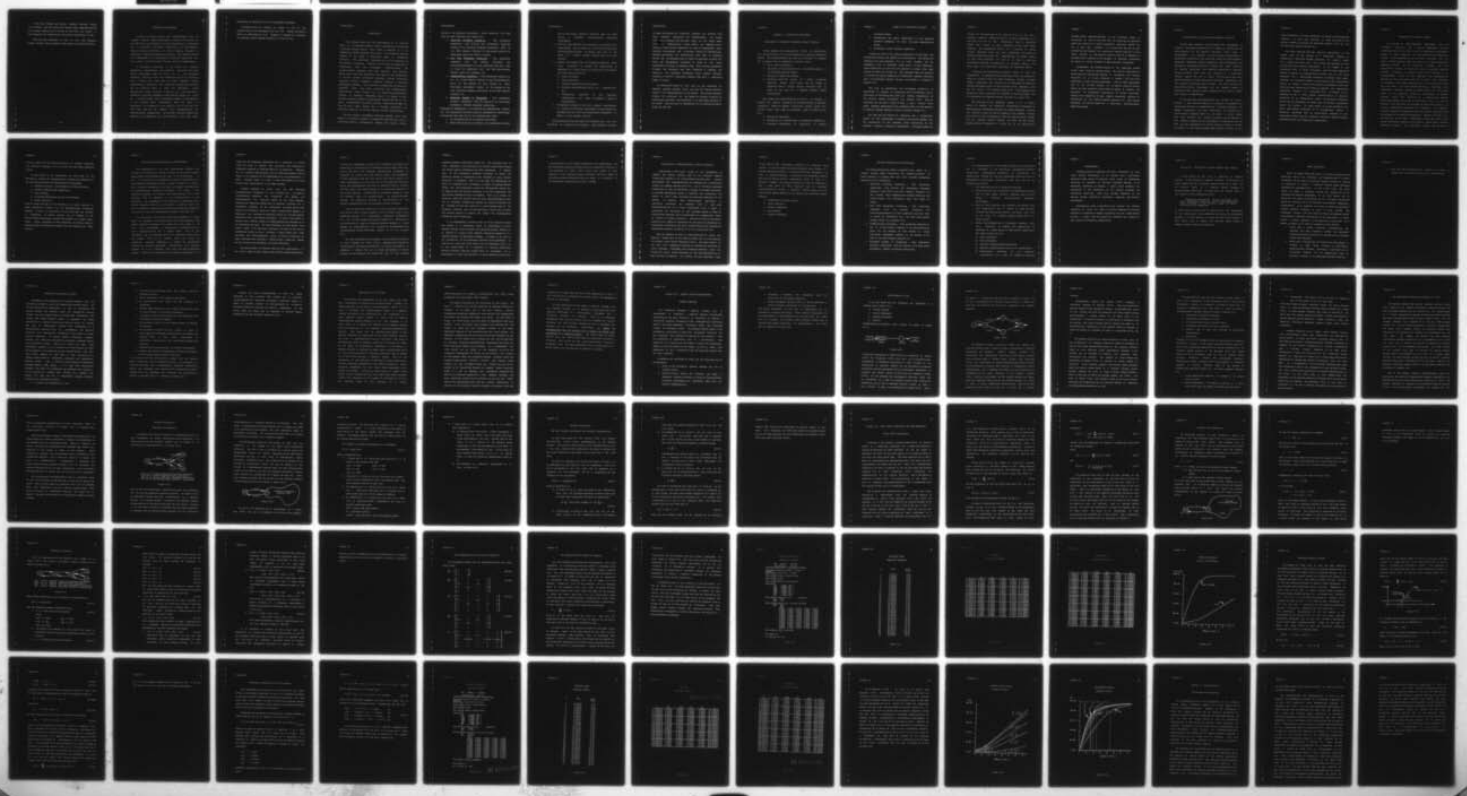
AD-A045 721 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/2
EQN MODELS FOR THE ANALYSIS AND DESIGN OF A COMPUTER NETWORK OF--ETC(U)
JUL 77 R L LEECH DAAG29-76-G-0309

UNCLASSIFIED ARO-14533.1-A-EL NL

1 OF 3
ADA
045721



School of
ENGINEERING
DUKE UNIVERSITY

EQN MODELS FOR THE ANALYSIS AND DESIGN OF A COMPUTER NETWORK OF FUNCTIONALIZED PROCESSORS

by

Robert L. Leech

Department of Electrical Engineering

Prepared Under:
U.S. Army Research Office
Grant No. DAAG29-76-G-309

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: (18) ARO (9) 14533.1-A-EL
2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle): (6) EQN MODELS FOR THE ANALYSIS AND DESIGN OF A COMPUTER NETWORK OF FUNCTIONALIZED PROCESSORS
5. TYPE OF REPORT & PERIOD COVERED: (9) Final rept. 15 Jul 76 — 30 Aug 77
6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s): (10) Robert L. Leech
8. CONTRACT OR GRANT NUMBER(s): (15) Grant No. DAAG29-76-G-0309

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Department of Electrical Engineering
Duke University
Durham, N. C. 27706

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Department of the Army
U.S. Army Research Office DRXRO-PR
P.O. Box 12211, Research Triangle Park, N.C. 27709

12. REPORT DATE: (11) 30 Jul 1977
13. NUMBER OF PAGES: 195 (12) 204p.

14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)

15. SECURITY CLASS. (of this report)

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES
The findings in this report are not to be construed as an official DA position unless so designated by other authorized documents.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computers	Command and Control
Computer Networks	Functionalized Processors
Queuing	Exponential Queuing Networks
Queuing Networks	Computer Architecture
Performance Evaluation	Computer Models
	Computer Simulation

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Network Queuing Theory is used to establish three Exponential Queuing Network (EQN) models of computer network systems. A projected computer network is based on the specification of processing functions for model representation. Example functions used in this work are computation, data base management, communications and terminal/access interface as would be applicable to a military command and control computer system. The models →

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

118230

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT (CONT'D.)

may be used for trade-off analysis of critical performance criteria. Additionally, a basic method for the synthesis/determination (design) of processor "computer power" to ensure a load balanced network is ascertained. The three models and their respective computer network evaluations for trade-off analysis are pursued with their inherent advantages and disadvantages. Additionally, these models are compared with a classical event driven simulation in order to demonstrate their internal consistency and to provide a basis for verification. All three models accommodate job routes (types) with one model providing exact solutions for each job type. These models accommodate large numbers of jobs so that an effective total computer network may be modelled. The sole restriction is the computational complexity. In order to implement the models, several computer program modules used for various solution stages of the models are provided. These modules include job typing or classes, solution of normalizing constants, and reduction to an equivalent network of two nodes.

This report was also a dissertation submitted in partial fulfillment of the requirements for the Degree of Doctor of Philosophy in the Department of Electrical Engineering in the Graduate School of Duke University, 1977.

EQN MODELS

for the

ANALYSIS AND DESIGN

of a

COMPUTER NETWORK OF FUNCTIONALIZED PROCESSORS

Robert L. Leech

Department of Electrical Engineering

Duke University

Date: July 30, 1977

Approved:

Kishor S. Trivedi, Chairman

Peter N. Marinos, Co-Chairman

Dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical Engineering in the Graduate School of Duke University

1977

ABSTRACT

EQN MODELS
for the
ANALYSIS AND DESIGN
of a
COMPUTER NETWORK OF FUNCTIONALIZED PROCESSORS

Robert L. Leech

Department of Electrical Engineering

Duke University

Date: July 30, 1977

Approved:

Kishor S. Trivedi

Kishor S. Trivedi, Chairman

P. N. Marinos

Peter N. Marinos, Co-Chairman

Louis W Nolte

Robert B. Kerr

An abstract of a dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical Engineering in the Graduate School of Duke University

1977

ABSTRACT

Network Queuing Theory is used to establish three Exponential Queuing Network (EQN) models of computer network systems. A projected computer network is based on the specification of processing functions for model representation. Example functions used in this work are computation, data base management, communications and terminal/access interface as would be applicable to a military command and control computer system. The models may be used for trade-off analysis of critical performance criteria. Additionally, a basic method for the synthesis/determination (design) of processor "computer power" to ensure a load balanced network is ascertained. The three models and their respective computer network evaluations for trade-off analysis are pursued with their inherent advantages and disadvantages. Additionally, these models are compared with a classical event driven simulation in order to demonstrate their internal consistency and to provide a basis for verification. All three models accommodate job routes (types) with one model providing exact solutions for each job type. These models accommodate large numbers of jobs so that an effective total computer network may be modelled. The sole restriction is the computational complexity. In order to implement the models, several computer program modules used for various solution stages of the models are provided. These modules include job typing or classes, solution of normalizing constants, and reduction to an equivalent network of two nodes.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS

I am deeply indebted for the thoughtful advice given me for this research by Dr. Kishor Trivedi and Dr. Peter Marinos. Dr. Trivedi has provided many insights and ideas at the inception and throughout the progress of this work. Dr. Marinos has provided both direction and concepts as well as inspiring me to complete my tasks in a timely and professional manner. Additionally, much gratitude is due to two of my colleagues, Dr. Sang Lee and Mr. Bruce Ballard, who have acted as sounding boards on many occasions, enabling me to better understand and define the problems encountered, and provided many beneficial suggestions concerning notations and procedures. Furthermore, the concerted efforts of Bruce Ballard to produce this dissertation under the UNIX text editing and formatting facility are appreciated.

Appreciation is acknowledged to Mr. Herb Goertzel and Mr. Mal Billings of the World Wide Military Command and Control (WWMCCS) ADP Management Division of the Joint Chiefs of Staff for their support and suggestions; to Dr. Jim Painter of the Command and Control Technical Center, Defense Communications Agency for his interaction and technical information concerning WWMCCS; and to Captain Sammy W. Pearson of Federal Simulations for his support in the use of the multiclass version of ASQ.

To my wife, Andrea, and family, Douglas, Matthew, Andrew and Bronwyn, who all helped and assumed added responsibilities in my stead, enabling me to devote my full time and effort to this research and dissertation, particular recognition is due.

This work was supported in part by U.S. Army Research Office, Durham, North Carolina under grant No. DA-29-76-G-0309.

NOTATIONAL CONVENTIONS

In order to provide clarity and correspondence with the computer program modules developed in support of this work and to facilitate the preparation of this dissertation, a particular set of recognized notational conventions has been adopted. Since the chosen conventions are those normally employed in scientific computer programming languages, such notation should be no impediment to the experienced, scientific reader and will be an asset to the experienced computer user and professional.

Of particular importance is the method for denoting subscripts, superscripts, and functions. Subscripts are given within parentheses such as p(i,j) for p_{ij}, and functions similarly denoted so that Tr(N) indicates that Tr is a function of N. Efforts have been made to avoid confusion between parenthetical subscripts and functions by explicitly pointing out the function names as they are developed. Where superscripts might normally be used as in p^k, the superscript indicator k has been appended to the function letter p to yield pk. As stated above, such a notational anomaly is one familiar to the computer user. Furthermore, where the order of operations in an equation is not explicit, the "*" and the "**" have been used as the symbols for multiplication and exponentiation, respectively. The standard operator symbols for addition (+), subtraction (-), and division (/) are also used.

Precedence of operators is as in programming languages.

Unordered sets are denoted by braces "{" and "}" and ordered sets by the delimiters "{|" and "|}". Vector and matrix names are underlined as in $\underline{P}$. Elements of vectors and matrices are enclosed within square brackets as in [23 45 43].

INTRODUCTION

This research deals with the establishment of an analytic model of a projected computer network employing the techniques of Network Queuing Theory. Such a model is referred to as an Exponential Queuing Network (EQN) Model. The projected computer network is one based on identified types of processing functions. The functions identified are at the discretion of the analyst or designer. However, in this work the functions of computation, data base management, communications, and terminal/access interface have been selected. The analytic models which have been established can assist in the formulation of various network structures employing defined processing functions through subsequent trade-off analysis of critical performance criteria. Three methods of evaluating the model are presented along with their advantages and disadvantages. Additionally, a basic method for the synthesis/determination (design) of processor "computing power" to ensure a load-balanced computer network is ascertained through the use of the model. Consequently, the models provide a quantitative means or tool for both a basic design and a thorough evaluation, or trade-off analysis, of various computer network organizations.

The four types of processing functions selected have been based on a projected concept of processing functions that can be identified within a geographical command and control system

suited to the military environment. These functions are seen from the user viewpoint and consist of:

1. Front-end Interface Processing - That processing associated with driving and interfacing terminals similar to the Terminal Interface Processors (TIP) of the ARPA Network [Roberts 73] and access control for authorized users and levels of usage.
2. Data Base Management Processing - That processing concerned with the update, retrieval, and creation/maintenance of files comprising the data base as would be associated with the data base machine concept ([ACM 77], [Canaday 74]).
3. Communications Processing - That processing required to act as a node within a network or the interconnection point for the passing of data traffic to other functional processors similar to the concept of the IMPs (Interfact Message Processors) of the ARPA Network [Roberts 73].
4. Principal Element of Processing - That processing normally associated with the ability to do high speed arithmetic, "number crunching" operations.

Geographical separation or co-location and combinations thereof for the processing functions could be determined by operational constraints such that any of the following may arise:

1. All functions may be in separate locations.
2. Where functions are co-located, the transmission medium

may be high speed parallel transfers such as those within a classical multi/parallel processing environment.

3. When all four functions are physically co-located, this organization may be viewed as a user function-oriented multiprocessing system. One might view this in the sense of a loosely coupled system (LCS) or a local network.
4. Numbers and capabilities of processing functions would vary. Selection of numbers and capabilities is dependent to a large degree on operational requirements and other factors such as:
   a. Fault tolerance.
   b. File allocation.
   c. Existing communication facilities.
   d. Existing organizational units, e.g., headquarters, etc.
   e. Hierarchical structure of the supported organization, e.g., chain of command or updating requirements.
5. Evaluation is based on the total network organization to include the relative performance of each functional processor and not just the communication subnetwork in support of the computer network.

The applications for the model as developed fall into the categories of analysis and of design. Three different methods

of model evaluation for trade-off analysis are pursued with their inherent advantages and disadvantages. The design technique is applied with the most basic of the three model evaluators. Additionally, these models are compared with a classical event driven simulation in order to demonstrate their internal consistency and to provide a basis for verification. Analysis and design are demonstrated such that an existing system may be parameterized in terms of functions to match the model and subsequently evaluated in terms of the major performance criterion of response time. With certain guidelines in mind, the architect may also formulate a topology and ascertain the necessary processing power (design process) required to achieve a specified response time with a specified number of users.

An additional result of this work is the provision of computer program modules which are used for various solution stages in the implementation of the models. The program modules include the provision for job typing or classes, solution of normalizing constants, and reduction to an equivalent network of two nodes. They may also be implemented on an interactive basis using the IBM TSO.

Chapter I: BACKGROUND DEVELOPMENT

Development of Parallel Processing Computer Systems

We may commence the developmental review by ascertaining the characteristics of a multiprocessing or parallel processing system. The characteristics may vary from designer to designer but in general are as enumerated in Enslow [1974]:

1. Two or more processors.
2. Operation of the processors in a coordinated manner at the hardware/software level.
3. Sharing of common main memory.
4. Operation under the control of a single integrated operating system which might even be viewed as operation under a single control structure such as would be the case for a computer network system organization.

With these four characteristics in mind it is natural to examine the general rationale for multiprocessing or parallel processing. There are several motivating factors which have brought about this type of processing. Some of these factors are:

1. Sharing of resources.
2. Increasing the computational or processing capability.
3. Providing concurrence of operations to achieve

increased speed.

4. Recognizing the speed limitations of the physical hardware prompting the need for other computational means.
5. Providing a fault tolerant capability.

Looking at each of the points enumerated, we find that the motivating factors have indeed been significant. The sharing of resources has been important due to their expense, and the consequent need to minimize these costs. It has, therefore, been impractical to allow any one resource to remain idle when another process can utilize it. The resources that are referred to include the entire spectrum of "classical" computer resources such as processors, memory, devices, and information [Madnick & Donovan 74].

The gain in computation and processing capability by permitting an overlap of operations and of processes, and in particular of servicing many users, is readily exemplified by the Multics system ([Organick 72], [Sekino 72]). This is certainly not the only example of such a gain. Nearly every major computer manufacturer has produced either or both general or special purpose systems in such a category [Enslow 74].

Not only has the sharing of resources had a significant impact on the development of parallel processing systems, but the recognition of the inherent speed limitations of the physical hardware devices is significant. Switching speeds of

devices are now operating at the level of 10 ** -10 sec and a commonly accepted limitation due to the heat dissipation is at the level of 10 ** -11 sec. Furthermore, even if switching speeds were reduced to zero, propagation delays would become dominant. Yet, propagation delays are a function of device circuit densities which are fast approaching the limits of optical resolution [Stone 75]. Consequently, we cannot expect to find much reduction nor will we be able to gain any significant increase in speed from the physical device. The need, then, is to find other imaginative or innovative means.

Certainly the replication of similar units, or even the distribution of processes among dissimilar units for execution, leads to a certain amount of fault tolerance. Typical straight forward approaches are achieved by the multiple and identical central processing units of such systems as the Honeywell 6000 and the Univac 1100 systems [Enslow 74]. More sophisticated systems for fault tolerance are achieved by various schemes for the networking of computer systems ([Cosell 75], [Doll 73]).

The structure of the operating system or of a control system to support the different architectural approaches to multi/parallel processing systems has, quite naturally, evolved and adapted to meet the varied demands of the systems. Perhaps this point is best illustrated by the IBM approach first through the LCS (loosely coupled system) and then the DCS (direct coupled system) [Freeman 76]. In the LCS it is questionable

whether actual multiprocessing, in the strictest sense, is occurring, in view of the fact that the criterion of operation under the control of a single integrated operating system are not in fact met. However, if we note that the two or more processors must operate in a controlled fashion relative to each other, similar to the network concept, then it is feasible to consider such a system in the category of parallel processing and hence in a broad category of multi/parallel processing.

Furthermore, the characteristics of the operating system itself have taken on various forms. The initial approaches of tying together cooperating modules to accomplish the various functions (as in OS/360 [Madnick & Donovan 74]) have been supplanted by more elegant approaches. Some of these approaches such as the RC-4000 [Hansen 73] or Hoare's monitor concept [Hoare 74] have provided a more positive means of control and assurance of cooperative interaction of system functions. Perhaps the culmination of such complex operating systems is represented by the Multics system [Organick 72]. This system represents the current epoch for a multi-user, multiprocessor operating system.

Identification of Specialized Computer Functions

We must note, however, that throughout the development of parallel processing systems there is always the recognition and identification of the specific functions which must be accomplished. For instance, we have already acknowledged the "classical" functions of the operating system for such systems - processor management, memory management, device management, and information management [Madnick & Donovan 74]. These functions are further subdivided into different mechanisms to implement various policies which have been valuable to a particular computer system and set of users. But even more than this, identifications of other specialized functions associated with the parallel environment have surfaced. Two such functions are associated with the requirement for the computer system to communicate with users or other computers and the need to handle vast amounts of information.

It has been well established that the concept of "front-ending" a main computer system by a computer to accomplish the specialized function of communications to include protocols and message preparation can most effectively be performed by a functionalized computer [Wood 73]. In effect it would appear that this approach very closely resembles the IBM concept of LCS. However, because of the requirement to communicate and the need for the transmission of various amounts of data to include the support of many users sending small amounts of data, it has

been determined to be more productive to employ a functionalized computer. In fact this approach is taken with many of the current systems to include the Honeywell Multics 6000 and the CDC 6000/7000 series [Enslow 74].

The other function that has received recognition in the recent past is the concept of a "back-end" computer or essentially a data-base machine ([Canaday 74], [Baum & Nsiac 76], [Marill and Stern 75]). This concept is analogous to that of the front-end computer development. In the former it was recognized that a large processing load was being generated solely to support the communications traffic and it was more efficient for a specialized processor to perform these tasks. Likewise, the increased traffic or processing loads encountered solely to access, maintain and retrieve information from data bases has been identified as being a candidate for specialized processing ([Aschim 73], [Maryanski 75]). In essence, we are beginning to see a move towards identification of specific functions to be accomplished within the parallel processing system and the assignment of these tasks to specific processors within the system. In other words, it is more effective for these specialized functions to be accomplished within a dedicated processor with an architecture tailored to the requirement rather than by a system of multiple, general purpose processors which are essentially homogeneous.

Networking of Computer Systems

The factors of functionalized development and the recognized limitation for a totally homogeneous parallel processing system to be able to meet the increasing requirements [Flynn 72] have supported the development of the networking concept. With networking we are no longer concerned that the entire network operate under the control of a single integrated operating system. We are concerned, however, with the cooperative sharing of resources among individual computer systems. In essence, "a computer network can be defined as an interconnected group of host computers that automatically communicate with one another and that can share such resources as programs, data bases, memory space, and the long-haul links with each other. In a true network situation, therefore, each constituent computer system can either operate in a local mode under its own operating system or participate in network activity under the direction of a higher-level network supervisor program, or it can do both" [Doll 73]. It can be observed that this does not necessarily imply that there is, or is not, a geographical separation of computers. The commonly accepted thoughts of networking were such that we generally took the systems to be physically separated. However, this may not, nor need not, be the case. Furthermore, we may note that in the network operation, control is now under a higher-level network supervisor program. The supervisor program may not need to

concern itself with the detailed control of special resources for specific purposes as is the case with the local operating system.

At this point it is interesting to note some of the motivating factors for networking and realize the similarity to the motivating factors for multi/parallel processing:

1. Resource sharing - particularly for files and data.
2. Greater computational capability.
3. Load sharing.
4. Increased efficiency and use of resources.
5. Fault tolerance.

Some of these factors, if not identical, are quite similar to those postulated earlier for the multi/parallel processing systems. In fact, at this point, we can state that the concept of networking is merely another way of looking at the multi/parallel processing concept but perhaps on a larger, macro scale. Being cognizant of this situation, we are now ready to examine those architectural changes that are occurring in these systems.

Architectural/Organizational Considerations

The introduction of low cost computational modules has brought forth much new or revised thinking in the area of system architecture and organization and hence in its functioning. In fact, it is difficult to separate what might be classified as a multi/parallel processing system, or even a network of several such systems and whether or not they are locally interconnected or geographically separated throughout the world. This situation, however, is no longer the primary factor as it is not germaine. What is essential is the identification of the functions, the assignment of the functions to processing units and the necessary control and interaction. Recently there has been a plethora of concepts and proposed architectures embracing the notion of low cost computational modules [IEEE Symp 77].

To note several of the more established examples we could begin by examining the structure of the Carnegie-Mellon C.mmp project ([Cohen & Jefferson 75], [Levin et al 75], [Wulf et al 75]). This architecture is essentially a crossbar structure with 16 processor units and 16 memory units. Each of the processors may be assigned any of the identified tasks within the system. There is no attempt, other than within the protection features themselves, to assign any specialized function to a specific processor. Consequently, this approach appears to be no different than what has been encountered earlier. Perhaps the large number of available processors to do

tasks and the lessening requirement for a processor to switch from one task to another once initiated and, therefore, a resulting lowering of overhead would be an exception. However, as the authors readily point out, the operating system itself, Hydra, is as complex and as comprehensive as any encountered to date. Another point to note is that this system is a local system which participates in the ARPA Network.

Another approach is taken with the BBN Pluribus Multiprocessor ([Bressler et al], [Cosell 75]). In terms of architectural concepts, the processors and memory are interconnected via multiple busses as the name implies. Although the operating system that is employed provides for a consistent treatment of all processors as equal units, some parts of the software, which are known to make heavy demands on resources, are provided as multiple copies in a private memory associated with a particular processor. In effect this seems to provide for a certain degree of functionalization although it is not identified as such. The main advantage of this system over others seems to be the fault tolerance, as there is not even a centralized switch to fail. Although it would appear that some processes are functionalized, there is still a generality of equal treatment that implies a sophisticated operating system for the control and management of system functions.

The macro-module or multiple function unit architecture is one which makes a more refined step towards functionalization.

At least the recognition is made of the possible utilization and advantages of identifying and implementing particular functions. Such is the case of the multiple function-unit processors as described by Bowra [1976] and Ornstein [1967] and implemented in such machines as the CDC 6600 and the IBM 360/91 [Enslow 74]. These approaches go to the other end of the spectrum since they assign the computational functions to specific units or modules and hence imply very little change in the operating system for a large scale machine. Instead they are concerned primarily with the hardware means of implementation. We cannot overlook, however, the associated problem of identification of the parallel computations and the necessary control criteria.

Lest one feel that these are the only structural approaches that may be taken we could examine a host of others such as ring structures [Farber 75], hierarchical structures [Aschenhurst 75], and virtual channel structures [Fraser 75]. Even some of the architects of C.mmp are studying systems composed of large numbers of microcomputers and clusters of microcomputers for multiprocessor system structures ([Jones et al 75], [Fuller 76]).

The most interesting and possible major differing structure is that proposed by Arden [1975]. Employing the concept of modularization and functionalization, "it should be possible to construct a system that is at least as powerful, is highly modular in construction, yet permits the use of much simpler

operating system structures" [Arden 75]. The limitation due to cost, complexity, and reliability of merely constructing bigger and faster central processors has been recognized. It appears that in the conceptual approach presented by Arden, the operating system is simplified by the transfer of many control functions to hardware. That is to say that "by identifying the state of a process with a processor, a number of process control tables and task dispatcher routes are obliterated" [Arden 75]. It appears that the control structure is in fact simplified and more easily associated with a processor. This is, in itself, an important trend and implies that the more functicnalization that can be achieved perhaps the simpler the control structure can be. Perhaps one can even arrive at a pcint whereby scme of the classical schemata may be applied [Karp & Miller 69] in order that correct control is assured and mcdels for deterministic evaluation may be developed.

It is interesting to observe that the trend described above may result in a simplified means cf implementing a total computer system, which includes the operating system, whether it is applied to a single local computer processing system or to a computer network of processing systems. In effect it is left to the definition of the processing function being performed. It could be on a very low level such as at the computation unit or at the higher-level cf a front-end or back-end (data-base machine) computer function, either local or networked. As a consequence, we have now arrived at a pcint whereby we have both

a reorientation of the system architecture and organization and the associated operating system from that typified by Multics to that presented by Arden [1975] which will result in more emphasis on the classical forms of schemata for both a means of controlling various functions and as a means of evaluation of the functional organization of such a system.

Utilization of Functionalized Digital Processors

Application of the above trends to the development of Command and Control Systems used by the National Command Authorities appears to be a particularly fertile area. Already development and experimentation are underway for a preliminary network of command systems [CCTC 76]. One of the major problems encountered is that of evaluating a system of network structures to satisfy requirements short of building and evaluating the system or even a long, involved and detailed simulation of the system. If, however, such architectural approaches as previously presented are examined hand in hand with functionalization, it is quite conceivable that a means of evaluating the structure of the proposed system in terms of quantitative measures can be ascertained on a comparative basis in respect to another structure. Through the use of analytical modelling as a means of evaluation we can gain the advantages of flexibility, relatively low cost, responsiveness and significant information capacity ([Coletta et al 74], [Kleinrock 75]).

When we examine the idea of functionalization within the parallel system what we are really looking for is the trade-off of hardware costs versus complexity costs. Even more subtle is the fact that complexity impinges on reliability and hence on fault tolerance. Presumably the processing hardware costs are constantly being driven downwards so that functionalization is more and more attractive. As a result, we must reevaluate some

of the factors that originally motivated the classical move towards the use of multi/parallel processing under control of a complex operating system. A primary factor for examination is that of resource sharing. In actuality the principal resource that must be shared from an operational point of view in a military command and control situation is that of information. Next a high level of fault tolerance must be achieved. Functionalization within a worldwide network seems to be the most obvious answer whereby we can achieve the following desired results:

1. Information resource sharing
2. Fault tolerance
3. Greater capability
4. Load sharing
5. Greater efficiency

Network Functions and Applications

The functionalization within a network lends itself to a natural network model definition and ensuing construct. At first analysis, it appears that four user oriented processing functions may be considered:

1. Front-end Interface Processing - That processing associated with driving and interfacing terminals similar to the TIPs of the ARPA Network [Roberts 73] but without their communication network functions and access control for authorized users and levels of usage.
2. Data Base Management Processing - That processing concerned with the update, retrieval, and creation/maintenance of files comprising the data base as would be associated with the data base machine concept ([ACM 77], [Canaday 74]).
3. Communications processing - That processing required to act as a node within a network or the interconnection point for the passing of data traffic to other functional processors similar to the concept of the IMPs of the ARPA Network [Roberts 73].
4. Principal Element of Processing - That processing normally associated with the ability to do high speed arithmetic, "number crunching" operations.

The interconnection of these user processing functions need not be dependent on any particular geographical/locality constraints. Geographical separation or co-location and combinations thereof for the four processing functions could be determined by operational constraints such that any of the following may arise:

1. All functions may be in separate locations.
2. Where functions are co-located, the transmission medium may be high speed parallel transfers such as exist within a classical multi/parallel processing environment.
3. When all four functions are physically co-located, then this organization may be viewed as a user function oriented multiprocessing system. One might view this in the sense of a loosely coupled system (LCS) or a local network.
4. Numbers and capabilities of processing elements would vary. Selection of numbers and capabilities is dependent to a large degree on operational requirements and other factors such as:
   a) Fault tolerance.
   b) File allocation.
   c) Existing communications facilities.
   d) Existing organizational units, e.g. headquarters.
   e) Hierarchical structure of the supported organization, e.g. chain of command or updating

requirements.

Although specific functions have been identified in this basic network discussion, it is obvious that a particular architect might desire to identify other types of processing functions to describe an entirely different network. It is desirable, therefore, to develop a model which provides for generality in the definition of the network topology and processing functions. What then becomes important is the network system processing parameters, topology and relative performance.

Consequently, such a generality will provide the desired objective of using the model to perform comparative tradeoff analysis of competitive design approaches and will additionally provide a basic tool for design and synthesis of a network to meet specific processing requirements.

Chapter II: EXPONENTIAL QUEUING NETWORK (EQN) MODELS

In this chapter we will look at modelling of computer systems as they pertain to computer networks. In particular we shall examine the use of network queuing theory as a basis for an analytic model of a computer network composed of functionalized processors. At this point one should keep in mind Frank's [1975] observation that:

> Modelling, analysis and design techniques have made substantial progress, but the best results are still obtained by the artistic blend of theory, experience, and pragmatism.

In other words, it should not be expected that the development of the models about to be presented will solve all anticipated problems. It will, however, be shown that the models provide a very useful tool to support the artistic blend needed in computer network design and analysis.

Model Objectives

First, we should note that there are various objectives for modelling which will influence any determination of a model. Basically, one may model an existing computer network or system to ascertain or obtain a projected behavior under different workloads or configurations. On the other hand, one might desire to model a projected system not knowing all of its detailed and specific components. It is this latter use of modelling for which our objective is set - to model a projected system not knowing all of its detailed and specific components. Consequently, it is important to be able to study the projected network system's behavior in order to gain insight and conceptual clarity into the overall network performance rather than perform a detailed examination of its operations. In other words, a global approach to the network is being made. As a result we expect to be able to accomplish three things:

1. Begin with a given, projected configuration and topology and then ascertain whether the projected network performs according to a desired set of criteria (trade-off analysis).
2. Begin with a desired set of criteria and then design a network to meet these criteria or performance specifications (design). Obviously, a practical mix of iterations between the two capabilities must be employed to arrive at an acceptable network solution.

3. Make the model mathematically tractable in order to perform the computations rapidly and inexpensively.

Simulation and Analytic Models

Throughout this presentation it becomes apparent that the intention has been to work with exponential queuing theory. Why not use conventional simulation? Or have we prejudiced the choice towards the analytic model and exponential queuing theory? The most common types of conventional simulation deal either with event-oriented models to represent the actual operations of a system as they occur event by event or through the use of empirically derived data represented in a deterministic, analytic form. As a consequence, the inherent criterion for conventional simulation is the desire to obtain precise knowledge. To obtain precise knowledge, however, implies that detailed information concerning a computer system to be modelled is known. Such is not always the case. This deficiency, however, can be tolerated if we are willing to limit our precision of the model as we have already stated. The type model which appears to work well in this situation is an analytic model based on stochastic processes and exponential queuing theory. Such a model is often termed an Exponential Queuing Network (EQN) model. It is very well established ([Basket and Muntz 73], [Bhandiwad and Williams 74], [Browne et al 75], [Buzen 75], [Buzen 76a], [Fuller 75], [Giammo 76], [Moore 71]) that the use of exponential queuing networks satisfies the following criteria:

1. Is simple and inexpensive to use.

2. May deal with situations where only partial workload knowledge exists.
3. May be used over a wide range of parameters.
4. Is particularly well suited for the isolation of parameters.
5. Accepts small changes in input values and does not lead to large changes in predicted performance.
6. Is several orders of magnitude more cost effective than conventional simulation models.
7. Can provide a range of quick answers useful in planning evaluations.
8. Can provide an insight into the network in terms of gross computer performance measures such as thruput and response time. In other words, qualitative or comparative quantitative and approximate answers are provided.
9. Predicts with some certainty the network performance.
10. Permits the follow-on development of arbitrary networks of the models under general conditions.

It is this set of EQN criteria which meshes with the desired model objectives and is therefore selected. In particular, it has been said that "as a mathematical approach, probability models are generally more realistic than deterministic models because they can represent the irregular and unpredictable demand by computer users." [Coffman and Denning 73]

Keeping the above considerations in mind, the model developed in this research will provide for an arbitrary interconnection of specified processing functions either to model an existing network so functionalized or to model a projected system not knowing the specific components. In the latter case the model may be emplcyed to provide design guidelines for the projected system as well.

Application of EQN Models

The validity and usefullness of the EQN models has been well established in practice and in publications. However, the models employed to date seem to be concentrated either in the area of analyzing the communications aspect of a computer network ([Kleinrock 74], [Kleinrock 76]), the evaluation of a local computer system ([Buzen 71], [Rose 76]) or locally interconnected systems [Browne et al 75]. The approach to be taken here is to examine the totality of the computer network from the standpoint of its complete processing/computational ability. This means that the communications aspect is just another subsystem (or subnetwork) of the total computer network and must be parameterized as part of the model. Once this is accomplished, there is complete similarity in viewing the model in its local or geographical separation. At this point, it becomes valuable to analyze the system from a hierarchical point of view to obtain values of specified parameters just as Browne et al [1975] have done for a specific system. This is the employment of the technique of the micro to macro evolution or of first examining subsystems of the computer network to obtain specified parameters and then using these parameters in the higher level model. In this way, some details of lower level models may be incorporated into the higher level model ([Basket & Muntz 73], [Browne et al 75], [Buzen 71], [Colella et al 74]). The approach taken in this research is to employ

functionalization as a means of incorporating the lower level parameters into the higher level network.

One common criticism of the EQN models is that they do not seem to relate to the actual operation of computer networks or systems. As we shall see in the next chapter, certain assumptions regarding exponential distributions and independence of servers and arrivals are known to be false. Yet this type of model, as noted above, has been shown to demonstrate very good results. Up to this point, most analysts have accepted the EQN models as tools and are extremely careful in the model employment because of the contradictions that exist between the necessary assumptions and the known operational variables in the systems. In 1971 Buzen [1971] pointed out that the EQN provided an entirely accurate representation, considering that only the most significant aspects were included. As a consequence, the model should not be judged on the goodness of fit of the exponential assumptions but rather on the validity and utility of the insights which are ultimately gained. However, Buzen has recently demonstrated ([1976a], [1976b]) that perhaps these results are not accidental after all. Through the use of the concept of an "Operational Method" of computer system analysis based on a set of concepts that correspond naturally and directly to observed properties of real computer systems, the same analytical results can be obtained as with the EQN. These results are encouraging since they are further indication, in addition to the excellent results obtained in practice, that the

approach to be taken with the use of EQN modelling is in fact a good one and may be accepted even though some of the assumptions may not be satisfied.

For the application of the model, it will be assumed that the computer network to be modelled is operating under peak workload conditions in a steady-state environment over a particular interval of interest. This is a reasonable assumption since one would expect that in a Command and Control situation the most critical possibility must be considered and designed for or evaluated. Furthermore, it cannot be overemphasized that our interest lies in gaining an insight into performance of a projected system. One cannot expect to obtain results that are more refined than the parameters provided to the model. Such is the case for a projected system. It must be recognized that the input parameters will be projections so we should expect only projections or insights as results.

Chapter III: NETWORK DEFINITION/STRUCTURE

General Approach

For discussion purposes a queuing network will be established to represent a computer network architecture consisting of the four functions as described in Chapter I (Front-end Interface Processing (FIP), Communications Processing (COMP), Data-Base Management Processing (DBM), and Principle Element of Processing (PEP)). The queuing network model is not necessarily restricted to these four processing functions but may represent any functions as defined by the architect. Only the complexity of computation will be a restriction. This complexity is a function of the number of nodes (or processors), number of jobs and number of job types. As will be seen, this complexity is not a limiting factor for practical models even for large networks.

Procedures are developed in order that the following may be accomplished:

1. Entry to the processing network through the FIP by terminal users.
2. Specify various routes for different job types to represent workload criteria in terms of functionalized processor requirements and assimilate them into the network representation.

3. Determine a response time expression used for evaluation of the network capacity.
4. Select processor service rates as a design procedure to obtain balanced utilization of the functions.

With these procedures we are able to accommodate the two problems of analysis and design. Three different approaches for the assimilation or representation of job types into the model will be obtained. As a result, three different models will be derived each having advantages and disadvantages. All three will be presented in this work.

Establishment of EQN

It is well known that the following are important in a queuing model [Fuller 75]:

1. Arrival process
2. Service mechanism
3. Queuing discipline
4. Routing

Schematically one denotes these factors as shown in Figure III-1.

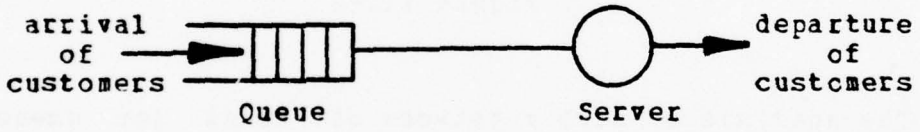


Figure III-1

A detailed explanation of these factors is contained in Fuller [1975] and Kleinrock [1975] and will not be repeated here. Furthermore, an extensive discussion of these factors in the formulation of specific models for the evaluation of quantum controlled service disciplines plus an extensive tabulation of research and survey papers in this area are included in Buzen [1972]. For our purposes the diagrammatic reference above may be considered as a node in a queuing network with a specified probability p(i,j) that a customer exiting a server of node i will proceed to node j as represented in Figure III-2. (Note:

In Chapter I a definition was given for a computer network and we are now speaking of a queuing network which is a model with more than one node and will be used to represent the computer network.)

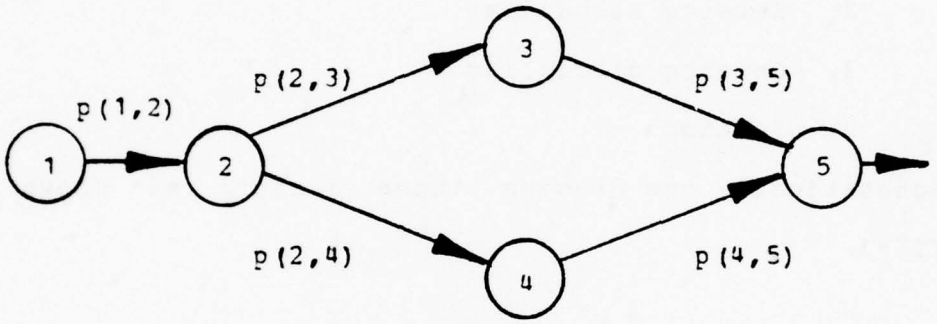


Figure III-2

The analysis of such a network of nodes (or queues) has been established over a number of years, culminating in specific algorithms for solution. Burke's Theorem ([Burke 56], [Kleinrock 75]) established that for an M/M/m queue in steady state condition with a Poisson input process (see Kleinrock [1975] for discussion of system categorization such as M/M/m, which represents a Poisson input process, an exponential service distribution, and m servers) the output is also Poisson and is independent of the other processes in the system. Jackson [1956] further established that for general networks of nodes, even with feedback, each node in the network behaves as though it were an M/M/m system with Poisson input. This is so even though the total input to a node is, in fact, not a Poisson

process.

Subsequently, Gordon and Newell [1967] examined a particular subcase of Jackson [1957]. They have considered queuing networks with completely general interconnecting paths as did Jackson but with the exception that only closed systems were examined. A closed system is one where there is no possibility of outside arrivals or departures of customers from the system. The closed queuing network systems are shown to be stochastically equivalent to open systems (those of Jackson) in which the number of customers is fixed to a specified value, say N.

The general solution for these networks is in the form of the probability of finding a specified number of customers at each of the nodes (queue/server pair) in the queuing network. Based on this solution, other performance measures of the network may be obtained to include the response time, utilization and thruput. The concept of local balance was introduced by Chandy [1972] to provide an easier method of solution. Local balance equates the rate at which customers enter and leave a single queue of a network, whereas global balance equates the rate at which a Markov process enters a state to the rate at which the process leaves the state. The work of Buzen [1973] and Chandy's study of the local balance technique are responsible for an efficient method of computing the performance measures of an EQN.

The mathematical basis for the classical queuing theory is quite well enumerated in the references cited in the previous paragraphs. However, as a summary, it should be noted that the following assumptions have been made in order to provide a mathematically tractable solution [Buzen & Denning 77]:

1. Stationary stochastic processes.
2. Stochastic independence of jobs.
3. Job transitions follow a Markov Chain.
4. Stochastic equilibrium has been achieved.
5. Service time at each node follows an exponential distribution.
6. Ergodicity.

Although this set of assumptions may not agree with the computer architect's concept of the actual computer system's operation, nor are the assumptions generally and readily understood by the computer system manager or the decision maker for system selection, the fact remains that the use of queuing models produces excellent results for performance analysis. Consequently, Buzen & Denning [1977] have proposed a different approach to queuing networks based on the concept of "operational analysis" [Buzen 76]. The basic premises employed are:

1. Finite observation periods. No assumption of steady state is made.
2. Work conservation. The number of entries to a given state must equal the number of exits from the state.

3. Homogeneity. The output rate at any node is dependent only on the number of jobs at that node.

This operational analysis approach yields the same solution forms as does the classical queuing theory ([Buzen 75], [Buzen 76a]). It would appear, however, that there is validity to the claim that this latter approach provides a higher correlation to actual computer system operation. Hence, it helps to make the modelling techniques employing queuing theory more readily acceptable.

A further extension to the theory which permits service time distributions other than exponential has been made by Chandy [1972]. He shows that when a processor sharing (PS) or last-come-first-served preemptive-resume (LCFSPR) discipline is assumed, the only relevant parameter in a service distribution with a rational Laplace transform is the mean value. Furthermore, a queue with either of the two conditions stated behaves the same way as a queue with a first-come-first-served discipline and exponential service distribution with the same mean. As a consequence, we may consider distributions other than exponential when we are interested in only the mean values. This fact allows more flexibility in network representation in the model. On the other hand, it also demonstrates the robustness of the queuing modelling technique since the solution forms have not changed. The modelling done in this work is restricted to the set of service disciplines and distributions just discussed.

ASQ (Arithmetic Solution to Queues) as a Tool

The solution methods for queuing networks provide closed form solutions but require considerable computation even with the techniques employed for the reduction of the computations. As a result, specialized computer programs have been developed for their implementation such as that of Moore [1971], the ASQ program ([Chandy 72], [Asplund 76]) and the QNET4 program [Reiser 76]. Each of these programs has specific limitations. The Moore [1971] program models the specific system then employed at the University of Michigan and accommodates one job type only. The Chandy ASQ program has been extended [Information Research Associates 75] to accommodate up to three job types and a total of 20 jobs and to operate in a general purpose mode. It is implemented in Fortran. The Duke version of ASQ [Asplund 76], also in Fortran, is a general purpose program model to accommodate up to 99 jobs of a single type. Finally, the QNET4 program, written in APL, is also of a general purpose nature with a limitation of two job types. Each of the programs mentioned (except Moore's) is proprietary and thus not available for general use.

One of the general program implementations using the principles and efficient computational methods discussed in the previous section has been developed by Chandy et al [1972] and further modified to operate in an interactive mode by Asplund [1976]. For the basic model development in Chapter IV, the

latter interactive implementation at Duke University known as ASQ (Arithmetic Solution to Queues) will be employed as a computational tool.

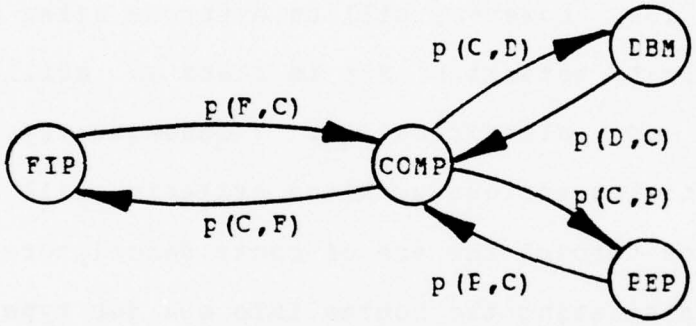
Duke ASQ provides a number of performance measures such as queue length at a node, utilization, thruput, and probability of the network being in a specified state. There is, however, no direct provision for determining the response time, Tr(N), as a function of the number of users in the network. (This limitation, however, will be overcome after a discussion of the preliminary network.) Nor is there an ability to handle job types or classification. Consequently, a procedure for discretizing various workload criteria will be developed and employed through the use of route descriptors. Then a procedure for assimilating the routes into one job type will be developed.

Two further model representations will be developed which do not rely entirely on the Duke ASQ and the use of single job types. Each of these models is developed with a set of programs which lend themselves to use on a general purpose basis. They have the ability to accommodate multiple job types and to compute response time directly for interactive computer network systems.

Network Description

Topology and Parameters

A construct for network definition is presented and will be used throughout for model determination and evaluation. The four functions of the selected network can be viewed to be topologically interconnected as in Figure III-3.



FIP (or F) denotes Front-end Interface Processing
COMP (or C) denotes COMmunications Processing
DBM (or D) denotes Data Base Management Processing
PEP (or P) denotes Principal Element of Processing

Figure III-3

Each of the nodes represents a queue/server pair with parameters for the type of processing function involved. In order to have an operational concept for this representation of a computer network as a queuing network, a projected and hopefully logical interpretation of processing functions was presented in Chapter I. At this time it should be recalled that the system architect or designer must go through similar analysis for the functional

representation of a computer network to be modelled. The four types of processing functions selected here, however, are based on an anticipated trend towards the four specified types of processing functions. We also believe that this is a logical architectural construct for a computer network.

The operational concept of this system is that jobs are submitted to the network by the users from interactive terminals. (The use of jobs is selected as a unit of work to be accomplished within the system. Depending on the architect's definition, the unit of work could be instructions, tasks, processes, customers, etc.) The terminals can be thought of as any type of input/output device but may be equated to the concept of a time-sharing terminal where the user alternatively thinks and initiates a job into the Computer Processing Network (CPN) ([Fuller 75], [Kleinrock 76], [Muntz and Wong 74]). These jobs circulate through the CPN and back to the terminal, forming a closed queuing network as shown in Figure III-4.

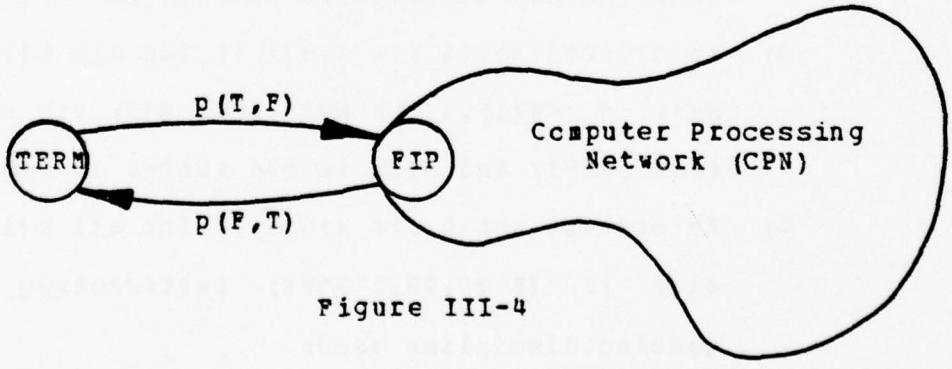


Figure III-4

The set of all terminals can be represented as a single node (TERM) and will be connected to the FIP of the Computer

Processing Network. The TERM node will consist of N servers representing N users. N is then indicative of the number of users active in the entire system (the terminals plus the Computer Processing Network, CPN) and will be a major factor in the design/evaluation process.

The network is formally represented as a quintuple

$$QN ::= \langle M, \underline{P}, S, D, R \rangle \qquad \text{(III-1)}$$

which is specified by:

1) A finite set M ::= {n(T),n(F),n(C),n(D),n(P),...} of nodes in the network such that

   n(T) ::= TERM n(D) ::= DBM

   n(F) ::= FIP n(P) ::= PEP

   n(C) ::= COMP

2) A matrix $\underline{P}$ ::= [p(i,j)] for all n(i),n(j) in M, where p(i,j) is the probability that a job leaving node n(i) will proceed directly to node n(j).

3) An ordered set S ::= {|s(i)|} for all n(i) in M, where s(i) = <u(i),a(i)> such that u(i) ::= service rate of each server and a(i) is the number of servers.

4) An ordered set D ::= {|d(i)|} for all n(i) in M, where d(i) in (FCFS,PS,LCFSPR) representing the type of queuing discipline used:

   FCFS - First Come First Served

   PS - Processor Sharing

   LCFSPR - Last Come First Serve Pre-emptive Resume

5) A finite set R ::= {r(k)}, where r(k) is an ordered pair composed of

   a) An ordered tuple <n(T),n(F),...,n(T)> traversing n nodes which we shall call a job route with each route representing a job type. Another view of the route is as a circuit in the network graph initiating at n(T) and terminating at n(T) with no intermediate occurrences of n(T). In our case, we will consider four routes, $1 \leq k \leq 4$. In theory there is just the practical computational limit on k.

   b) The frequency of occurrence (normalized to 1), q(k), of route r(k).

Network Description

Job Type (Route) Sub-Network and Terminal Representation

At this stage there are two further items that require clarification before actual implementation of the queuing network. The first of these points is the representation of the k (in this instance k=4) permissible job types or routes into the single network and the second is the situation of the TERM node.

In order to represent the various job routes, we form a k-th subnetwork for each distinct r(k) as a quadruple. (Note that the k corresponds to the k-th route and is appended as a subscript for the subnetwork and as a superscript for the elements of the quintuple.)

$$SN(k) ::= \langle Mk, \underline{Pk}, Sk, Dk \rangle \qquad \text{(III-2)}$$

which is specified by:

1) A finite set Mk ::= {n(i)} of nodes in the subnetwork. Note that the following necessary condition must hold in order that topological integrity be maintained:

$$Mk \subset (\text{the first element of } r(k)) \subseteq M \qquad \text{(III-3)}$$

2) A matrix $\underline{Pk}$ ::= [pk(i,j)] for n(i) and n(j) in Mk, where pk(i,j) is the probability that a job leaving

node n(i) will proceed directly to node n(j) for the k-th route.

3) An ordered set Sk ::= {|s(i)|} for all n(i) in Mk, where s(i) = <u(i),a(i)> such that u(i) ::= service rate of each server and a(i) is the number of servers. Note that the following necessary condition holds:

$$Sk \supseteq S \qquad \text{(III-4)}$$

Furthermore, the service rate at a processor s(i) is not a function of the route, but is constant over all routes. We shall see in later model development the relaxation of this condition.

4) An ordered set Dk ::= {|d(i)|} for all n(i) in Mk, where d(i) is in (FCFS,PS,LCFSPR). Again note that the following necessary condition holds:

$$Dk \supseteq D \qquad \text{(III-5)}$$

The TERM is structured such that s(T) ::= <u(T),N). By our terminology Z is the mean think time of a user at a terminal or, in other words, the mean time between completion of a user's job and his subsequent initiation of another job. As a result, with i terminals ($0 \leq i \leq N$) in the thinking mode, the effective service rate for the TERM node is

$$u(T) = f(i) = i / Z \qquad \text{(III-6)}$$

Note that the terminal node can be treated as an Infinite

Service (IS) station and, therefore, no queuing occurs at the node. This structure for TERM then yields the desired result and is the same approach as that established in Kleinrock [1967, 1976] and Muntz and Wong [1974].

Chapter IV: BASIC MODEL FORMULATION AND IMPLEMENTATION

Basic Model Formulation

Previous to this chapter a general description of queuing theory as a modelling technique and a means of providing a network description has been presented. We are now ready to establish the first of the three models to be presented which we will call the Basic Model (BM). The model approach is to find a means of consolidating or assimilating the k job types into one. Consolidating job types into one can reduce the computational complexity and make it possible to use the available ASQ program [Asplund 76] at Duke University which simplifies the implementation of the problem solution. Such a procedure is reported in Leech [1976]. The consolidation of job types is done by assuming that superposition of the k subnetworks into one queuing network may be accomplished.

The procedure for assimilation of the k routes and their respective k subnetworks into one queuing network is accomplished by noting from equations III-1 and III-3 the elements M, S, and D may be formed by the ordered merging of Mk for all k into M, Sk for all k into S and Dk for all k into D. This merging implies the restriction that s(i) and d(i) are identical for all routes regardless of their occurrence in a particular route. R must be specified by ascertaining the r(k)

$\forall$ k. What remains to be determined in equation III-1 is the probability matrix $\underline{P}$. First however, $\underline{P}k$ $\forall$ k must be determined. The method for determining $\underline{P}k$ is applicable not only to the basic model but will be used for succeeding models and later references. First it is noted that the information about the route is in the form of a deterministic description. We have to obtain the subnetwork transition probability matrix from this description. The following technique can be used for this purpose.

Define tk(i,j) to be the number of <n(i),n(j)> adjacent pairs occurring in the first element of r(k). These adjacent pairs define the arcs in the route. Next ascertain the number of times that a job traverses or visits a node vk(i) which is

$$vk(i) = \sum_j tk(i,j) \qquad \text{(IV-1)}$$

Now the proportion of the job visits that leave n(i) to go to n(j) is

$$pk(i,j) = tk(i,j) \;/\; vk(i) \qquad \text{(IV-2)}$$

This procedure then completely defines the $\underline{P}k$ $\forall$ k.

In order to evaluate $\underline{P}$ from the $\underline{P}k$ $\forall$ k, the individual elements p(i,j) of $\underline{P}$ are evaluated based on the probability pk(i,j), the i-th and j-th element of $\underline{P}k$, which has been calculated, conditioned on the occurrence of the k-th route, r(k). The probability that route k, r(k), occurs is q(k).

Therefore,

$$p(i,j) = g(i) \sum_{k} pk(i,j) * q(k), \quad \text{for } n(i),n(j) \text{ in } (M,Mk) \qquad \text{(IV-3)}$$

(Recall that the appendage of k denotes a superscript over which the summation occurs.)

where

$$g(i) = 1 / \left(\sum_{k} I(i,k) * q(k) \right) \qquad \text{(IV-4)}$$

and

$$I(i,k) = \begin{cases} 1 & \text{if } n(i) \text{ is in } r(k) \\ 0 & \text{otherwise} \end{cases} \qquad \text{(IV-5)}$$

The assumption made here is that the q(k), defined as the likelihood of the occurrence of the k-th route, is also the likelihood for the occurrence of a job at the k-th route at a particular node (normalized based on the node being in the k-th route). This is a reasonable assumption if the amount of time that a job remains in the computer processing network is much greater than the think time. That is to say, Tr(N) >> Z. Put in other words, this means that the CPN is loaded to the maximum and nearly no jobs are thinking. Such an approach should provide the worst case situation in regards to response time or an upper bound. The point to be ascertained is what inaccuracies are incurred and to what extent. Such a comparison for the selected network will be presented in Chapter V.

Response Time Expression

As discussed earlier, we must determine a means of evaluating the mean response time, Tr(N), as a function of the number of users in the total system. The response time is defined as the time that a job is initiated into the network until the job is returned to the terminal from the network. Fortunately, an extremely useful relationship is available to us. Little's Result [Kleinrock 76] states that:

$$J = L * T \qquad \text{(IV-6)}$$

where J ::= number of jobs in the computer network system

L ::= mean arrival rate of jobs to the computer network system

T ::= mean time in the computer network system

It is the mean time in the system that we have defined as the response time. If we know the number of jobs in the terminal node, say Q(T), then J = N - Q(T). Now recall the general configuration of the network in Figure IV-1 (same as Figure III-4):

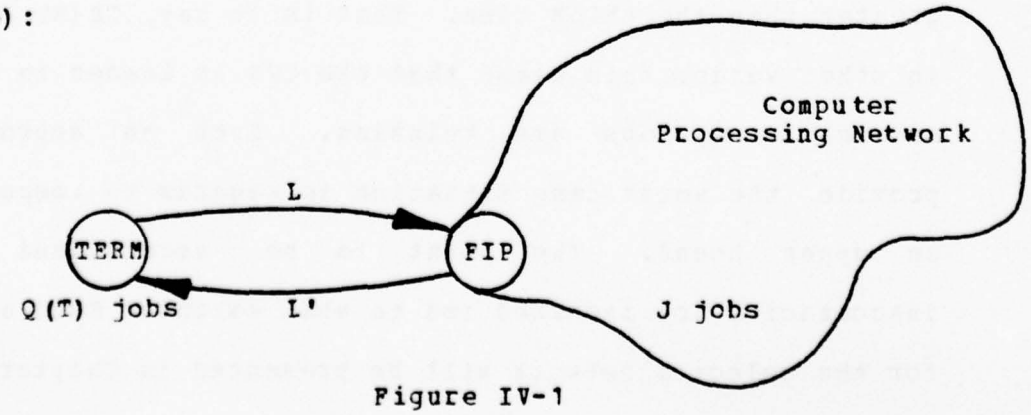


Figure IV-1

By Little's result, equation IV-6 becomes

$$L' = Q(T) / Z \qquad \text{(IV-7)}$$

which is the input to the terminal node coming from the rest of the network and

$$L = J / Tr(N) \qquad \text{(IV-8)}$$

which is the output from the terminal node going to the rest of the network. Also since equation III-2 holds such that no queue is allowed to form, then in the steady state condition

$$L = L' \qquad \text{(IV-9)}$$

Consequently, combining IV-7 and IV-8 into IV-9:

$$Q(T) / Z = J / Tr(N)$$

or re-arranged

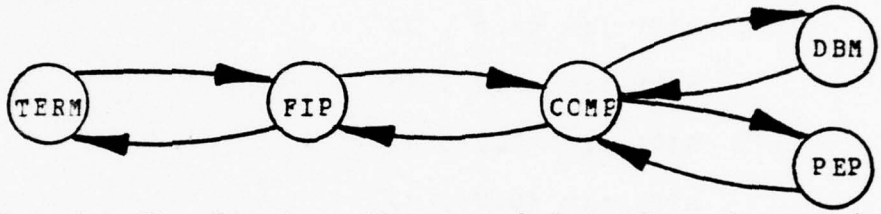
$$Tr(N) = (N - Q(T)) Z / N(T) \qquad \text{(IV-10)}$$
$$= [N / Q(T) - 1] Z$$

which is the desired result. In the latter expression it may be noted that Tr(N) is inversely related to Q(T), the queue length or number of jobs at TERM, which is the only parameter being varied at this point. This condition is supported by intuitive reasoning. The fewer the number of jobs at the terminal (or in a think state) the greater is the number of jobs being

processed, since this is a closed network. With a larger number of jobs in the processing state or loading the Computer Processing Network, the longer it can be expected for the jobs to be processed.

Parameter Definition

With the augmentation of the Terminal node (TERM) as in Figure III-4 the network previously shown in Figure IV-1 is shown in Figure IV-2.



FIP (or F) denotes Front-end Interface Processing
COMP (or C) denotes COMmunications Processing
DBM (or D) denotes Data Base Management Processing
PEP (or P) denotes Principal Element of Processing

Figure IV-2

This network according to our structure is represented as

$$QN ::= \langle M, P, S, D, R \rangle \qquad \text{(IV-11)}$$

with the following element representations:

1. $M ::= \{n(T), n(F), n(C), n(D), n(P)\}$ (IV-12)

   where

   $n(T) ::= TERM$ $\qquad$ $n(D) ::= DBM$

   $n(F) ::= FIP$ $\qquad$ $n(P) ::= PEP$

   $n(C) ::= COMP$

2. The matrix P is unknown at this point and must be calculated based on the P_k as described earlier in this chapter.

3. $S ::= \{|s(T), s(F), s(C), s(D), s(P)|\}$ (IV-13)

which does not provide distinctions between routes for this model. The number of servers at a node and the service rates of these servers are specified as follows:

s(T) ::= <1/Z , N> (IV-14)

s(F) ::= <1.5 , 1> (IV-15)

s(C) ::= <1.0 , 1> (IV-16)

s(D) ::= <0.2 , 1> (IV-17)

s(P) ::= <0.2 , 1> (IV-18)

The u(i) have been initially selected as though these are known mean service rates for existing or projected processors to perform each of the functions.

4. D ::= {|PS, FCFS, FCFS, PS, PS|} (IV-19)

The d(i) are selected based on the same rationale as the u(i). That is, the selection is based on existing or projected processors to perform each of the functions. Again, distinction between routes is not possible for the basic model.

5. R ::= {r(1), r(2), r(3), r(4)} (IV-20)

Four routes have been selected as being representative of a possible ordering of processing functions to be performed for the four different job types:

a. r(1) ::= (<n(T), n(F), n(T)> ; 0.6) (IV-21)

This route may be visualized as one job type requiring query generation processing or menu selection, or even security access. In other

words, the short interactive requests that would be serviced within a "local" geological area to the FIP. We expect a large percentage of the total number of requests to be of this type. Consequently, q(1) is expected to be rather large.

b. r(2) ::= (<n(T), n(F), n(C), n(D), n(C), n(F), n(T)> ; 0.2) (IV-22)

This route is representative of a data base update or retrieval requirement with no large scale computation involved as no demands are made to the PEP.

c. r(3) ::= (<n(T), n(F), n(C), n(D), n(C), n(P), n(C), n(F), n(T)> ; 0.1) (IV-23)

This route is representative of a retrieval or update followed by a subsequent requirement to perform considerable processing prior to job return to the user.

d. r(4) ::= (<n(T), n(F), n(C), n(P), n(C), n(F), n(T)> ; 0.1) (IV-24)

This route represents a strictly computational job, i.e. not requiring access to the DBM.

It is important to realize at this point that the definition of routings and associated probabilities is part of the parameter selection and to a great extent is derived from the designers' or analysts' artistic blend of theory, experience, and pragmatism discussed in Chapter I. Values

selected in this implementation are representative of a possible configuration and are in no way intended to justify a particular system.

ASQ Implementation for Trade-Off Analysis

The parameters defined for the implementation of the Basic Model yield:

$$\underline{P1} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad \text{(IV-25)}$$

$$\underline{P2} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{(IV-26)}$$

$$\underline{P3} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 & 0 \\ 0 & 1/3 & 0 & 1/3 & 1/3 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \text{(IV-27)}$$

$$\underline{P4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \\ 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \text{(IV-28)}$$

$$\underline{P} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0.8 & 0 & 0.2 & 0 & 0 \\ 0 & 0.45833 & 0 & 0.33333 & 0.20833 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \text{(IV-29)}$$

ASQ Calculation for Trade-Cff Analysis

Once the parameter definition and determination have been completed, it is relatively straight fcrward to introduce these values into Duke ASQ following the ASQ Users' Manual [Asplund 76]. For the preceeding values the ASQ Network Description is at Figure IV-3. It should be noted that the u(T) is calculated in accordance with equation III-2 and is based on Z = 15 seconds. Values for $1 \leq N \leq 75$ are provided. The Z is obtained based on the reasoning that r(1) jobs, by virtue of their description, should have a mean think time equal to five seconds as where the other jobs, r(k), $2 \leq k \leq 4$, should have a mean think time equal to thirty seconds. Weighting the think times in accordance with their respective probabilities of occurrence, we have produced a virtual mean think time as follows:

$$Z = \sum_{k} zk\ q(k) \qquad \text{(IV-30)}$$

where zk is the think time for route k. Note that the assumption of maximum loading or that all jobs are in the CPN is also made here as was done for determining q(k).

At Table IV-1 are the computed results for the Basic Model as defined. Shown is the queue length at n(T), Q(T), and the respective response time function Tr(N) in accordance with equation (IV-10). Additionally, the thruput and utilization of each functional processing node may be easily computed using ASQ (Tables IV-2 and IV-3, respectively). Curves for the Tr(N) and

utilization for the processor with the largest values(DBM) are also shown in Figure IV-4. This data thus provides information regarding the overall network performance and is but one iteration of an evaluation process. It is evident that additional iterations may be performed varying the different parameters to obtain a relative comparison of the network performance under various conditions.

A noteworthy point at this juncture is the utilization for DBM of Table IV-3. For this particular selection of the u(i), the DBM acts as the bottleneck as defined in Muntz and Wong [1974]. The next bottleneck occurs with the FIP followed by the PEP and then the COMP. Or, in other words, we can increase the u(F), u(P), and u(C) and will not improve the response time or thruput as long as the DBM remains the bottleneck. Jobs just simply cannot proceed through the network any faster. This observation, consequently, further illuminates the desire for a design/synthesis procedure.

NETWORK DESCRIPTION

TRADE-OFF ANALYSIS

```
          ASQ    VERSION 7    RELEASE 3

           DUKE UNIVERSITY   JUNE 1976
**********************************************

THE NETWORK DESCRIPTION RESIDES %!<FILENAME> ?
TRADEOFF
ENTER NUMBER OF JOB TYPES (1 OR MORE)
          1
ENTER NUMBER OF NODES + BRANCHING POINTS
          5
ENTER NUMBER OF NODES
          5
WILL YOU HAVE EXTERNAL POISSON SOURCES?
          NO
ENTER EDGES
          FIP      0.20000  COMP ;
          FIP      0.80000  TERM ;
          COMP     0.45834  FIP  ;
          COMP     0.33333  DBM  ;
          COMP     0.20834  PEP  ;
          TERM     1.00000  FIP  ;
          DBM      1.00000  COMP ;
          PEP      1.00000  COMP ;
          #
ENTER RATES FOR NODES WITH FCFS          DISCIPLINE
          FIP     1.50
          COMP    1.00
          #
ENTER RATES FOR NODES WITH PS OR LCFSPR DISCIPLINE
          DBM     0.20
          PEP     0.20
                  TERM
                 0.06667   0.13333   0.20000   0.26667   0.33333
                 0.40000   0.46667   0.53333   0.60000   0.66667
                 0.73333   0.80000   0.86667   0.93333   1.00000
                 1.06667   1.13333   1.20000   1.26667   1.33333
                 1.40000   1.46667   1.53333   1.60000   1.66667
                 1.73333   1.80000   1.86667   1.93333   2.00000
                 2.06667   2.13333   2.20000   2.26667   2.33333
                 2.40000   2.46667   2.53333   2.60000   2.66667
                 2.73333   2.80000   2.86667   2.93333   3.00000
                 3.06667   3.13333   3.20000   3.26667   3.33333
                 3.40000   3.46667   3.53333   3.60000   3.66667
                 3.73333   3.80000   3.86667   3.93333   4.00000
                 4.06667   4.13333   4.20000   4.26667   4.33333
                 4.40000   4.46667   4.53333   4.60000   4.66667
                 4.73333   4.80000   4.86667   4.93333   5.00000
                  #
ENTER DEGREE(S) OF MULTIPROGRAMMING
                   1

THE TEMPLATE IS:

FIP  COMP DBM  PEP  TERM
```

Figure IV-3

RESPONSE TIMES

TRADE-OFF ANALYSIS

N	Q(T)	Tr(N) (sec)
1	0.84004	2.86
2	1.66889	2.98
3	2.48554	3.10
4	3.28890	3.24
5	4.07782	3.39
6	4.85102	3.55
7	5.60716	3.73
8	6.34480	3.91
9	7.06241	4.12
10	7.75838	4.33
11	8.43108	4.57
12	9.07884	4.83
13	9.69999	5.10
14	10.29290	5.54
15	10.85605	5.73
16	11.38808	6.07
17	11.88782	6.45
18	12.35437	6.85
19	12.78717	7.29
20	13.18600	7.75
21	13.55106	8.25
22	13.88299	8.77
23	14.18279	9.33
24	14.45189	9.91
25	14.69198	10.52
30	15.53124	13.97
35	15.96338	17.89
40	16.18793	22.06
45	16.31142	26.38
50	16.38304	30.78
55	16.42639	35.22
60	16.45313	39.70
65	16.47008	44.20
70	16.48073	48.71
75	16.48756	53.23

Table IV-1

THRUPUT

TRADE-OFF ANALYSIS

N	FIP	COMP	DBM	PEP	N
1.00000	0.07001	0.03055	0.01018	0.00636	1.00000
2.00000	0.13907	0.06069	0.02023	0.01264	2.00000
3.00000	0.20713	0.09038	0.03013	0.01883	3.00000
4.00000	0.27408	0.11960	0.03987	0.02492	4.00000
5.00000	0.33982	0.14828	0.04943	0.03089	5.00000
6.00000	0.40425	0.17640	0.05880	0.03675	6.00000
7.00000	0.46726	0.20390	0.06797	0.04248	7.00000
8.00000	0.52873	0.23072	0.07691	0.04807	8.00000
9.00000	0.58853	0.25682	0.08560	0.05350	9.00000
10.00000	0.64653	0.28212	0.09404	0.05878	10.00000
11.00000	0.70259	0.30659	0.10219	0.06387	11.00000
12.00000	0.75657	0.33014	0.11005	0.06878	12.00000
13.00000	0.80833	0.35273	0.11757	0.07349	13.00000
14.00000	0.85774	0.37429	0.12476	0.07798	14.00000
15.00000	0.90467	0.39477	0.13159	0.08225	15.00000
16.00000	0.94901	0.41411	0.13804	0.08628	16.00000
17.00000	0.99065	0.43229	0.14409	0.09006	17.00000
18.00000	1.02953	0.44925	0.14975	0.09360	18.00000
19.00000	1.06560	0.46499	0.15499	0.09688	19.00000
20.00000	1.09883	0.47949	0.15983	0.09990	20.00000
21.00000	1.12925	0.49277	0.16425	0.10266	21.00000
22.00000	1.15691	0.50484	0.16828	0.10518	22.00000
23.00000	1.18190	0.51574	0.17191	0.10745	23.00000
24.00000	1.20432	0.52552	0.17517	0.10949	24.00000
25.00000	1.22433	0.53425	0.17808	0.11131	25.00000

Table IV-2

UTILIZATION

TRADE-OFF ANALYSIS

N	FIP	COMP	DBM	PEP	N
1.00000	0.04667	0.03055	0.05091	0.03182	1.00000
2.00000	0.09272	0.06069	0.10114	0.06322	2.00000
3.00000	0.13808	0.09038	0.15064	0.09415	3.00000
4.00000	0.18272	0.11960	0.19933	0.12459	4.00000
5.00000	0.22655	0.14829	0.24714	0.15447	5.00000
6.00000	0.26950	0.17640	0.29400	0.18376	6.00000
7.00000	0.31151	0.20390	0.33983	0.21240	7.00000
8.00000	0.35249	0.23072	0.38453	0.24034	8.00000
9.00000	0.39236	0.25682	0.42802	0.26753	9.00000
10.00000	0.43102	0.28213	0.47020	0.29389	10.00000
11.00000	0.46839	0.30659	0.51097	0.31937	11.00000
12.00000	0.50438	0.33014	0.55023	0.34391	12.00000
13.00000	0.53889	0.35273	0.58788	0.36744	13.00000
14.00000	0.57183	0.37429	0.62381	0.38990	14.00000
15.00000	0.60311	0.39477	0.65794	0.41123	15.00000
16.00000	0.63267	0.41412	0.69019	0.43139	16.00000
17.00000	0.66043	0.43229	0.72047	0.45031	17.00000
18.00000	0.68635	0.44925	0.74875	0.46799	18.00000
19.00000	0.71040	0.46499	0.77498	0.48438	19.00000
20.00000	0.73255	0.47949	0.79915	0.49949	20.00000
21.00000	0.75284	0.49277	0.82127	0.51332	21.00000
22.00000	0.77128	0.50484	0.84139	0.52589	22.00000
23.00000	0.78793	0.51574	0.85956	0.53725	23.00000
24.00000	0.80288	0.52553	0.87587	0.54744	24.00000
25.00000	0.81622	0.53426	0.89042	0.55654	25.00000
30.00000	0.86285	0.56478	0.94128	0.58833	30.00000
35.00000	0.88685	0.58049	0.96747	0.60470	35.00000
40.00000	0.89933	0.58866	0.98109	0.61321	40.00000
45.00000	0.90619	0.59315	0.98857	0.61788	45.00000
50.00000	0.91017	0.59576	0.99291	0.62060	50.00000
55.00000	0.91258	0.59733	0.99554	0.62223	55.00000
60.00000	0.91407	0.59830	0.99716	0.62325	60.00000
65.00000	0.91500	0.59891	0.99818	0.62389	65.00000
70.00000	0.91560	0.59931	0.99883	0.62430	70.00000
75.00000	0.91598	0.59956	0.99925	0.62456	75.00000

Table IV-3

TRADE-OFF ANALYSIS

Tr(N) & UTILIZATION

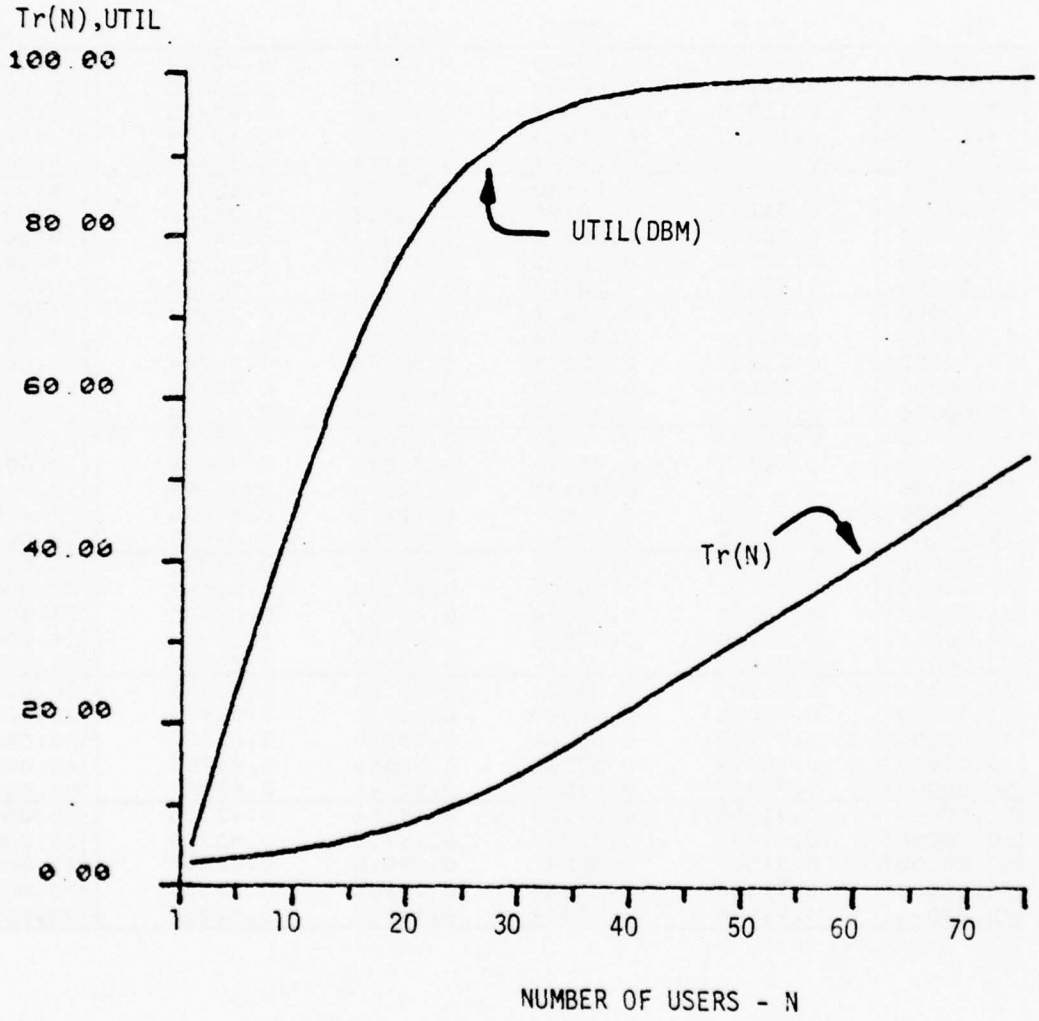


Figure IV-4

Balanced Processor Design

Now suppose we would like to have the most efficient processor utilization, which implies an optimum thruput rate, or at least know what it should be for a specified network. We do know that such a network should be a balanced one ([Moore 71], [Maekeura and Boyd 71]). This does not imply that what results is the most efficient network in terms of other parameters. It does indicate where efficiency may be achieved. Therefore, we would like to be able to select service rates to ensure a balanced system [Muntz and Wong 74]. In terms of our structure QN = <M, $\underline{P}$, S, D, R> we need to determine S. However, let us consider the case of a(i) = 1 $\forall$ i since we know [Kleinrock 76] that for any s(i) = <u(i),a(i)> we can obtain an equivalent s(i,eq) = <u(i,eq),1>. Therefore, we will assume a(i) = 1 in this discussion. Furthermore, we can now proceed in the development of the u(i,eq) $\forall$ n(i) so that all nodes approach saturation uniformly and no one node becomes a bottleneck. Hence we will have a balanced system. Based on the work of Muntz and Wong [1974] we have the following asymptotic properties for Tr(N) as N becomes large:

$$Tr(N) = N\,(v(s) / u(s)) - Z \qquad \text{(IV-31)}$$

Here we let

$$v(i) = L(i) / L(T) \qquad 2 \leq i \leq |M| \qquad \text{(IV-32)}$$

where v(i) is the average number of visits to node n(i) per user request as before and I(i) is the mean arrival rate of jobs at node i. T denotes the TERM node as before. The subscript s denotes the saturation node n(s). This asymptotic condition is shown in Figure IV-5. The line for Tr(1) is also shown and is given by

$$Tr(1) = \sum_{k} v(k) / u(k) \qquad \text{(IV-33)}$$

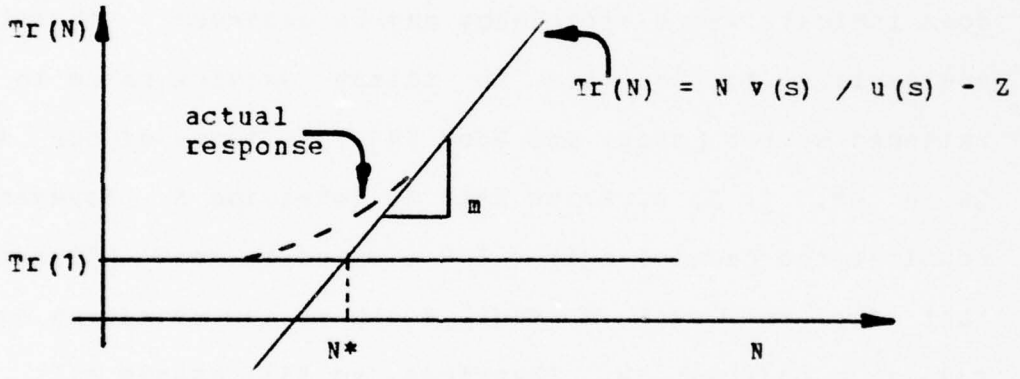


Figure IV-5

As N becomes large then from equation IV-31 the slope m of the linearized response time is expressed as

$$m = v(s) / u(s) \qquad \text{(IV-34)}$$

Since saturation is being approached at the same rate for all nodes, in a balanced system one has

$$v(i) / u(i) = m \quad \forall\, n(i),\ i \neq T \qquad \text{(IV-35)}$$

Using IV-35 in IV-31 and IV-33 we have

$$Tr(N) = N\,m - Z \qquad \text{(IV-36)}$$

$$Tr(1) = (|M| - 1)\,m \qquad \text{(IV-37)}$$

Equations IV-36 and IV-37 may be equated in order to solve for N*, the point of intersection of the two asymptotes given by

$$N^* = ((|M| - 1)\,m + Z) / m \qquad \text{(IV-38a)}$$

such that

$$m = Z / (N^* - |M| + 1) \qquad \text{(IV-38b)}$$

We can observe that IV-35 may be rewritten in the form

$$u(i) = v(i) / m \quad \forall\, n(i), \quad i \neq T \qquad \text{(IV-39)}$$

Thus, if we are given the acceptable level of response time, Tr(1), equation IV-37 can be used to evaluate m. Alternatively, if we are given the number of terminals that must be supported without saturating the computer network, then equation IV-38b can be used to evaluate m. Consequently, if we should know v(i), then equation IV-39 may be employed to provide a means of deriving the desired service rates, u(i). But recall that IV-32 does establish the v(i). We merely need to solve for the L(i) and let L(T) = 1 and the desired u(i) are determined. To solve for L(i) we may employ the balance equations of Gordon and Newell [1967] for the entire network such that:

$$v(i) = \sum_{j} v(j)\; p(i,j) \quad n(i),n(j) \text{ in } M \qquad \text{(IV-40)}$$

The L is not uniquely defined but by fixing the L(T) = 1 for all n(i) where $i \neq T$ the L can then be uniquely determined.

Synthesis Procedures for Load Balancing

The complementary capability for utilization of the Basic Model, as previously described, is that of a synthesis procedure to achieve the most "efficient processor utilization." We have said that this should be done by selecting processor service rates so that all processing nodes approach saturation uniformly with no one node becomming a bottleneck.

Employing the procedure just described, we first proceed to obtain the v(i) by use of equation IV-40 with v(T) = 1:

$$[\,1, v(F), v(C), v(D), v(P)\,] = [\,1 \;\; v(F) \;\; v(C) \;\; v(D) \;\; v(P)\,]\,\underline{P} \qquad \text{(IV-41)}$$

where $\underline{P}$ is given in equation IV-29. At Appendix I is a PL/I program which solves the v(i) given the $\underline{P}$ as input. This program uses a routine for the solution of M equations in N unknowns of Gallie and Ramm [1976] and was provided by Ballard and Sigmon [1977]. Using the Appendix I program we obtain the following:

$$v(F) = 1.2500$$

$$v(C) = 0.54546$$

$$v(D) = 0.181813$$

$$v(P) = 0.113638$$

Assuming a saturation of N* = 24 is tolerable, then by IV-38b we have

$$m = Z / (N^* - |M| + 1) = 15 / (24 - 5 + 1) = 0.75 \quad \text{(IV-42)}$$

and by equation IV-37 it is seen that

$$Tr(1) = (|M| - 1) = 4\ (0.75) = 3.0 \text{ seconds} \quad \text{(IV-43)}$$

This is the horizontal asymptote of Figure IV-5 which may be assumed to be an acceptable value. Consequently the u(i) are:

$$\begin{aligned} u(F) &= 1.25 / 0.75 = 1.66667 && \text{(a)} \\ u(C) &= 0.54546 / 0.75 = 0.72728 && \text{(b)} \\ u(D) &= 0.181813 / 0.75 = 0.24241 && \text{(c)} \\ u(P) &= 0.113638 / 0.75 = 0.15152 && \text{(d)} \end{aligned} \quad \text{(IV-44)}$$

The network description for the balanced implementation governed by equations IV-11 and IV-41 is at Figure IV-8. Table IV-4 lists the response times which is followed by the thruput and utilization tables, IV-5 and IV-6, respectively.

NETWORK DESCRIPTION

BALANCED SYSTEM

```
          ASQ    VERSION 7    RELEASE 3

           DUKE UNIVERSITY   JUNE 1976
**************************************************

THE NETWORK DESCRIPTION RESIDES *!<FILENAME> ?
BALANCED
ENTER NUMBER OF JOB TYPES (1 OR MORE)
          1
ENTER NUMBER OF NODES + BRANCHING POINTS
          5
ENTER NUMBER OF NODES
          5
WILL YOU HAVE EXTERNAL POISSON SOURCES?
          NO
ENTER EDGES
          FIP       0.20000  COMP ;
          FIP       0.80000  TERM ;
          COMP      0.45834  FIP  ;
          COMP      0.33333  DBM  ;
          COMP      0.20834  PEP  ;
          TERM      1.00000  FIP  ;
          DBM       1.00000  COMP ;
          PEP       1.00000  COMP ;
          #
ENTER RATES FOR NODES WITH FCFS          DISCIPLINE
          FIP     1.66667
          COMP    0.72728
          #
ENTER RATES FOR NODES WITH PS OR LCFSPR DISCIPLINE
          DBM     0.24241
          PEP     0.15152
                  TERM
                 0.06667   0.13333   0.20000   0.26667   0.33333
                 0.40000   0.46667   0.53333   0.60000   0.66667
                 0.73333   0.80000   0.86667   0.93333   1.00000
                 1.06667   1.13333   1.20000   1.26667   1.33333
                 1.40000   1.46667   1.53333   1.60000   1.66667
                 1.73333   1.80000   1.86667   1.93333   2.00000
                 2.06667   2.13333   2.20000   2.26667   2.33333
                 2.40000   2.46667   2.53333   2.60000   2.66667
                 2.73333   2.80000   2.86667   2.93333   3.00000
                 3.06667   3.13333   3.20000   3.26667   3.33333
                 3.40000   3.46667   3.53333   3.60000   3.66667
                 3.73333   3.80000   3.86667   3.93333   4.00000
                 4.06667   4.13333   4.20000   4.26667   4.33333
                 4.40000   4.46667   4.53333   4.60000   4.66667
                 4.73333   4.80000   4.86667   4.93333   5.00000
                  #
ENTER DEGREE(S) OF MULTIPROGRAMMING
                  1

THE TEMPLATE IS:

FIP  COMP DBM  PEP  TERM
```


Figure IV-6

RESPONSE TIMES

BALANCED SYSTEM

N	Q(T)	Tr(N) (sec)
1	C.83332	3.00
2	1.65517	3.12
3	2.46459	3.26
4	3.26058	3.40
5	4.04212	3.55
6	4.80809	3.72
7	5.55732	3.89
8	6.28863	4.08
9	7.00077	4.28
10	7.69251	4.50
11	8.36261	4.73
12	9.00986	4.98
13	9.63312	5.24
14	10.23131	5.33
15	10.80349	5.83
16	11.34888	6.15
17	11.86685	6.49
18	12.35701	6.85
19	12.81916	7.23
20	13.25339	7.64
21	13.65998	8.06
22	14.03950	8.51
23	14.39270	8.97
24	14.72056	9.46
25	15.02423	9.96
30	16.22632	12.73
35	17.02826	15.83
40	17.57477	19.14
45	17.96167	22.58
50	18.24664	26.10
55	18.46381	29.68
60	18.63420	33.30
65	18.77121	36.94
70	18.88358	40.60
75	18.97751	44.28

Table IV-4

THRUPUT

BALANCED SYSTEM

N	FIP	COMP	PEP	DBM	N
1.00000	0.06945	0.03030	0.01010	0.00631	1.00000
2.00000	0.13793	0.06019	0.02006	0.01254	2.00000
3.00000	0.20538	0.08962	0.02987	0.01867	3.00000
4.00000	0.27172	0.11857	0.03952	0.02470	4.00000
5.00000	0.33684	0.14699	0.04900	0.03062	5.00000
6.00000	0.40067	0.17484	0.05828	0.03643	6.00000
7.00000	0.46311	0.20209	0.06736	0.04210	7.00000
8.00000	0.52405	0.22868	0.07623	0.04764	8.00000
9.00000	0.58340	0.25457	0.08486	0.05304	9.00000
10.00000	0.64104	0.27973	0.09324	0.05828	10.00000
11.00000	0.69688	0.30410	0.10136	0.06336	11.00000
12.00000	0.75082	0.32763	0.10921	0.06826	12.00000
13.00000	0.80276	0.35030	0.11676	0.07298	13.00000
14.00000	0.85261	0.37205	0.12401	0.07751	14.00000
15.00000	0.90029	0.39286	0.13095	0.08185	15.00000
16.00000	0.94574	0.41269	0.13756	0.08598	16.00000
17.00000	0.98890	0.43152	0.14384	0.08990	17.00000
18.00000	1.02975	0.44935	0.14978	0.09362	18.00000
19.00000	1.06826	0.46615	0.15538	0.09712	19.00000
20.00000	1.10445	0.48194	0.16065	0.10041	20.00000
21.00000	1.13833	0.49673	0.16557	0.10349	21.00000
22.00000	1.16996	0.51053	0.17017	0.10636	22.00000
23.00000	1.19939	0.52337	0.17446	0.10904	23.00000
24.00000	1.22671	0.53529	0.17843	0.11152	24.00000
25.00000	1.25202	0.54634	0.18211	0.11382	25.00000

Table IV-5

UTILIZATION

BALANCED SYSTEMS

N	FIP	COMP	DBM	PEP	N
1.00000	0.04167	0.04167	0.04167	0.04167	1.00000
2.00000	0.08276	0.08276	0.08276	0.08276	2.00000
3.00000	0.12323	0.12323	0.12324	0.12323	3.00000
4.00000	0.16303	0.16303	0.16304	0.16303	4.00000
5.00000	0.20211	0.20211	0.20212	0.20211	5.00000
6.00000	0.24040	0.24040	0.24042	0.24041	6.00000
7.00000	0.27787	0.27787	0.27788	0.27787	7.00000
8.00000	0.31443	0.31443	0.31445	0.31443	8.00000
9.00000	0.35004	0.35004	0.35006	0.35004	9.00000
10.00000	0.38463	0.38463	0.38465	0.38463	10.00000
11.00000	0.41813	0.41813	0.41815	0.41813	11.00000
12.00000	0.45049	0.45049	0.45052	0.45050	12.00000
13.00000	0.48165	0.48165	0.48168	0.48166	13.00000
14.00000	0.51156	0.51156	0.51160	0.51157	14.00000
15.00000	0.54017	0.54017	0.54021	0.54018	15.00000
16.00000	0.56744	0.56744	0.56748	0.56745	16.00000
17.00000	0.59334	0.59334	0.59338	0.59335	17.00000
18.00000	0.61785	0.61785	0.61789	0.61786	18.00000
19.00000	0.64096	0.64096	0.64099	0.64096	19.00000
20.00000	0.66267	0.66267	0.66271	0.66267	20.00000
21.00000	0.68300	0.68300	0.68304	0.68300	21.00000
22.00000	0.70197	0.70197	0.70201	0.70198	22.00000
23.00000	0.71963	0.71963	0.71967	0.71964	23.00000
24.00000	0.73603	0.73602	0.73607	0.73603	24.00000
25.00000	0.75121	0.75121	0.75125	0.75121	25.00000
30.00000	0.81132	0.81132	0.81136	0.81132	30.00000
35.00000	0.85141	0.85141	0.85146	0.85142	35.00000
40.00000	0.87874	0.87874	0.87879	0.87875	40.00000
45.00000	0.89809	0.89809	0.89814	0.89809	45.00000
50.00000	0.91233	0.91233	0.91238	0.91234	50.00000
55.00000	0.92319	0.92319	0.92324	0.92320	55.00000
60.00000	0.93171	0.93171	0.93177	0.93172	60.00000
65.00000	0.93856	0.93856	0.93862	0.93857	65.00000
70.00000	0.94419	0.94418	0.94424	0.94419	70.00000
75.00000	0.94883	0.94887	0.94893	0.94888	75.00000

Table IV-6

As is expected, $Tr(1) = 3.0$ which is in accord with equation IV-43. Furthermore, Table IV-6 does verify that the selection of the $u(i)$ for all $n(i)$, $i \neq T$, does in fact provide a totally balanced system as the utilization factor is the same for each processor for all N. Figure IV-7 shows the asymptotic values and the $Tr(N)$ from the model for various value of m. It is obvious that as m is reduced the horizontal asymptote drops and the large N linearization asymptote, equation IV-36, is reduced in slope. Consequently, a corresponding improvement is gained for all N and the N* is increased as well. However, as shown in Figure IV-8, a corresponding decrease in utilization is encountered as it should be. What is most interesting, however, is that for a corresponding N* and m, $0.375 \leq m \leq 1.33$, there is a difference of less that 10 percent in the balanced utilization. Furthermore, this point is occurring near the knee of the curve, indicating that N* may be thought of as the optimum load.

RESPONSE TIME CURVES

BALANCED SYSTEM

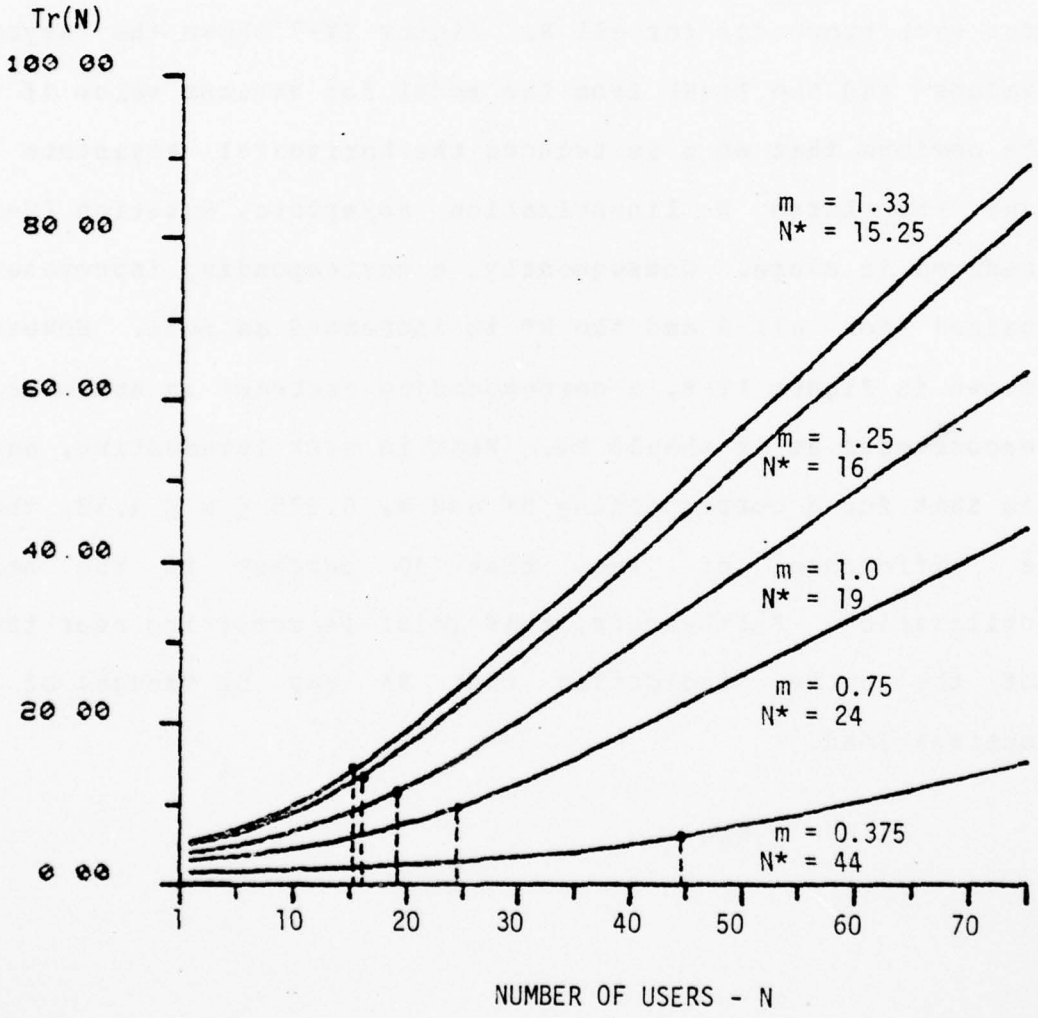


Figure IV-7

UTILIZATION CURVES

BALANCED SYSTEM

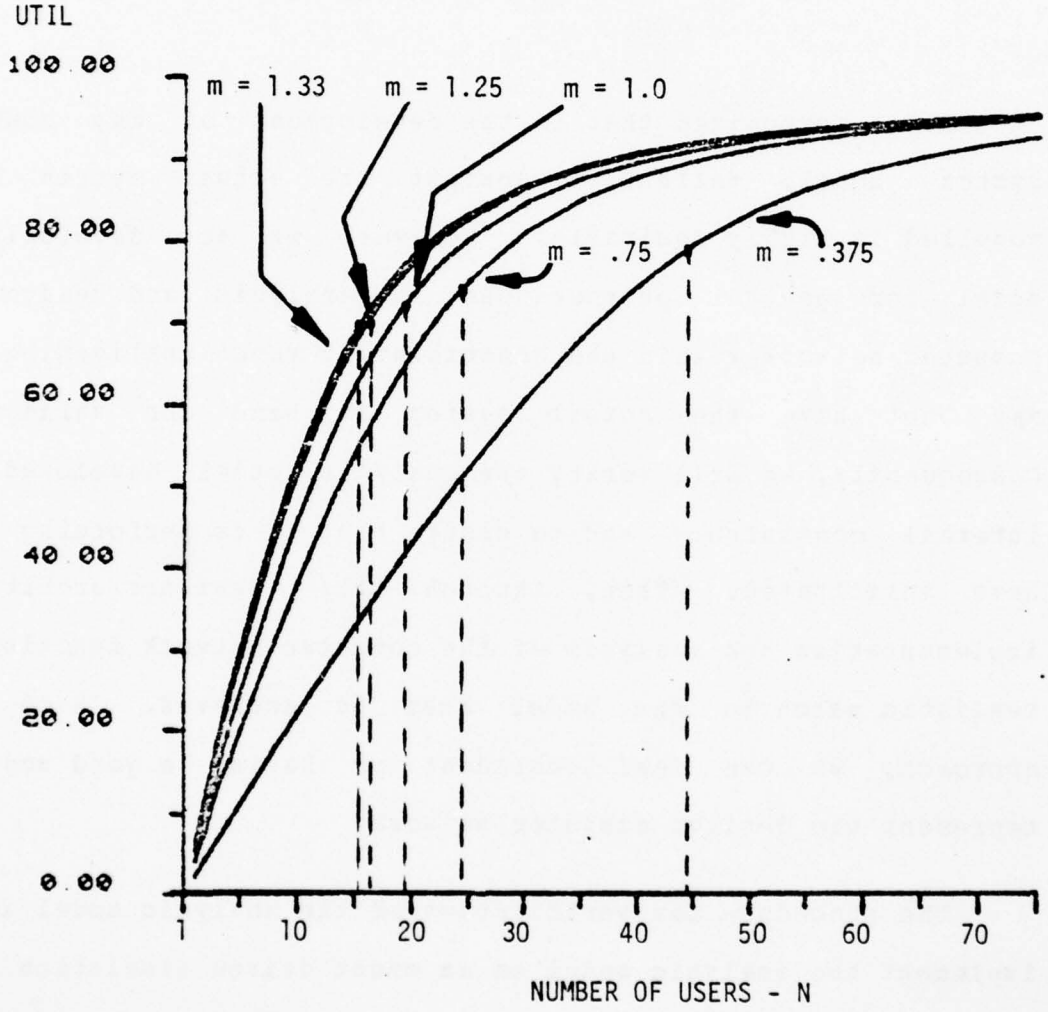


Figure IV-8

Chapter V: GPSS SIMULATION

Discussion and Description

It is recognized that in the development of any computer system model, validation against the actual system being modelled is highly desirable. However, we are developing a model for general purpose use in analysis and design of a computer network within the constraint of functionalization. We may not have the actual system at hand for validation. Consequently, we will verify the analytic model developed for internal consistency and to ensure that it is performing as we have anticipated. Then, through the designer/architect's implementation and analysis of the computer network functions, a realistic match to the model must be achieved. With this approach, we can feel confident of having a good model to represent the desired computer network.

The procedure for verification of the analytic model is to implement the analytic model as an event driven simulation model and compare the results based on our primary performance criterion of mean response time. The simulation system selected is the General Purpose Simulation System (GPSS) [IBM 71]. This system was selected because of its well established use for event driven simulation of computer systems ([Colella et al 74], [Schriber 74]), its ready availability, the convenience of use

at the higher, macro level and the ability to readily control desired parameters.

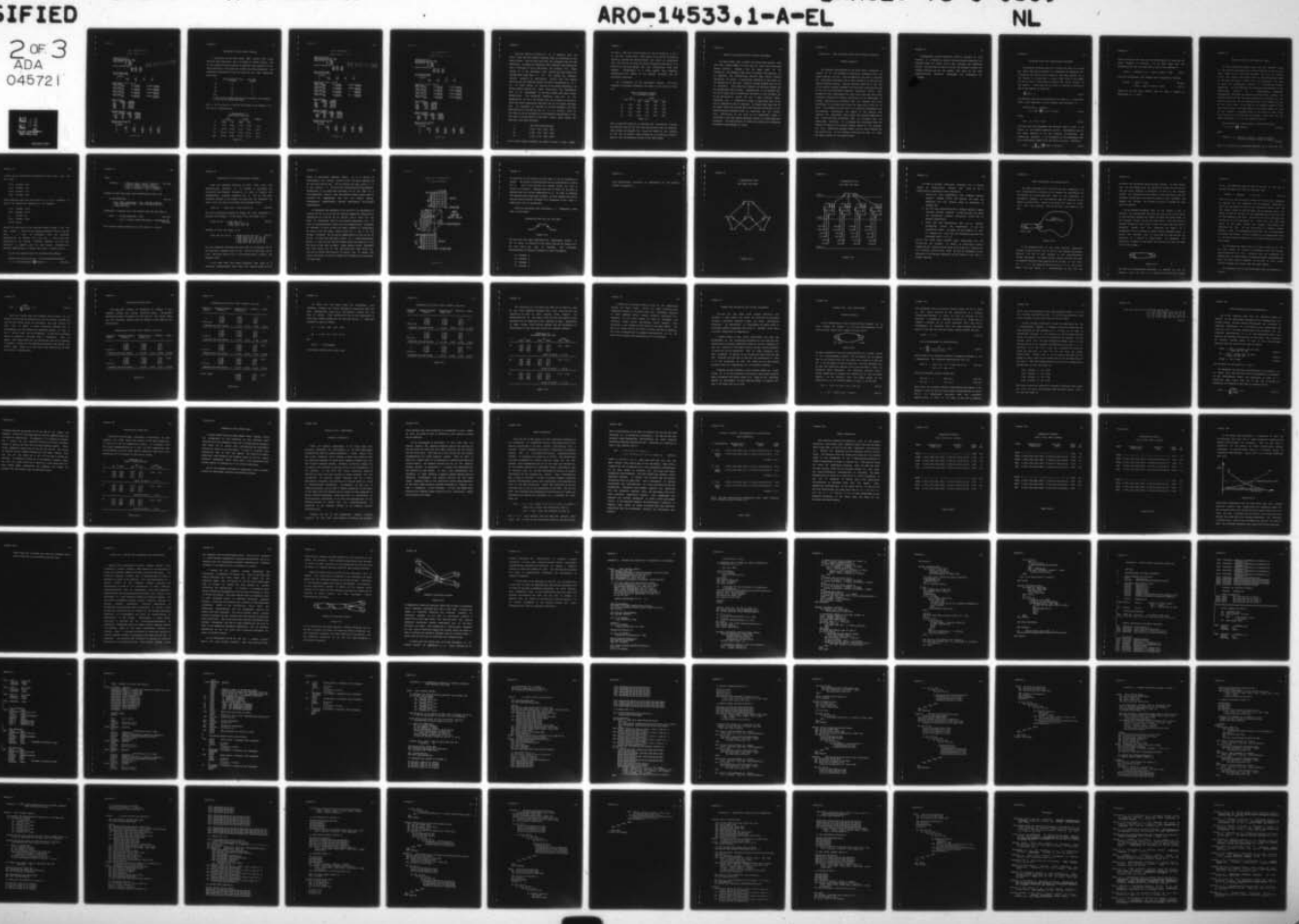
The representation and implementation in GPSS of the network as determined in Chapter III is included in Appendix B. In this GPSS simulation, exact deterministic routings as described for r(k), $1 \leq k \leq 4$, are implemented, provisions for other service time distributions are made, and the capability for a much more detailed report generation is available. Although exponential service distributions are used for exact comparison to the EQN model, other distributions can be used in GPSS. Additionally, we will be able to readily note the effects of our assumptions used in assimilating the k routes into one. A sample of a typical GPSS output is at Figure V-1. It is easy to see in this figure some of the additional information provided by the simulation compared to that provided by the Basic Model calculations of Chapter IV. First, general information concerning the simulation run is tabulated. At this point, it should be noted that an initialization of the simulation run is required prior to data collection [Schriber 74]. This initialization is necessary to force the simulation into a steady state condition. Other data is the mean think times zk for each route with a k correspondence of {|1,2,3,4|} ::= {|A,B,C,D|}. The mean service times for each function and the type of distribution, in this case exponential, are listed. This is followed by information concerning both the queue and processor statistics. Next a matrix showing the absolute count

on row one and the relative frequency (normalized to 1000) on row two of the k job types (columns) entering the CPN is displayed. This relative frequency is the job load distribution in the CPN. The mean composite response time for all job types and each job type's mean response time (RESRk) and their respective standard deviations are enumerated (in milliseconds). This data is gathered from tables maintained by GPSS which may also be individually listed, if desired, as well as a probability density function plot of the response times. Lastly, a matrix, processor entry count by routes, showing the number of visits of each job type (columns 1-4) to each node (rows 1-5), is tabulated. Note that these values compare to the vk(i) with an i correspondence of {|1,2,3,4,5|} ::= {|T,F,C,D,P|}. The fifth column is the sum of all jobs entering a functional processor.

AD-A045 721 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/2
EQN MODELS FOR THE ANALYSIS AND DESIGN OF A COMPUTER NETWORK OF--ETC(U)
JUL 77 R L LEECH DAAG29-76-G-0309

UNCLASSIFIED ARO-14533.1-A-EL NL

2 OF 3
ADA
045721



GPSS SIMULATION OF

BASIC MODEL, N = 10


```
NUMBER OF JOBS IS  10
INITIALIZATION HOURS ARE   6.00
DATA COLLECTION HOURS ARE   2.00
THINK TIMES ARE - A:  15000
                  B:  15000
                  C:  15000
                  D:  15000
SERVICE TIMES(MSEC) ARE -
                        FIP:  EXP    600
                        COMP: EXP   1375
                        DBM:  EXP   4125
                        PEP:  EXP   6600

THE JOB COUNT BY ROUTE
 FULLWORD MATRIX RTCNT

      ROW/COLUMN      1          2          3          4

       1           2718        559        198        251
       2            729        150         53         67


COMPOSITE MEAN R TIME IS  4317.410
 TABLE  RESR1
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
            2718              891.017                   899.000
 TABLE  RESR2
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
             559            10747.160                  5360.000
 TABLE  RESR3
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
             198            21322.492                  9440.000
 TABLE  RESR4
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
             251            13686.769                  8544.000


QUEUE STATISTICS
 QUEUE      AVERAGE      AVERAGE
          CONTENTS     TIME/TRANS
 TERMQ       7.765      14970.199
  FIPQ        .582        885.714
 COMPQ        .603       1962.601
  DBMQ        .543       5166.402
  PEPQ        .504       8078.222


PROCESSOR STATISTICS
FACILITY   AVERAGE     NUMBER     AVERAGE
         UTILIZATION  ENTRIES    TIME/TRAN
    FIPF      .394       4734      599.330
   COMPF      .437       2215     1421.429
    DBMF      .425        757     4048.403
    PEPF      .412        450     6595.597


PROCESSOR ENT COUNT BY ROUTE
 FULLWORD MATRIX   ENT

      ROW/COLUMN      1          2          3          4          5

       1           2718        559        198        251       3726
       2           2718       1118        396        502       4734
       3              0       1118        594        503       2215
       4              0        559        198          0        559
       5              0          0        198        251        449
```

Figure V-1

Comparison to Basic Model Results

A comparison of the Basic Model (BM) results with those from the GPSS simulation (S) is at Table V-1 which follows. Due to the relatively large amount of computer time required for the GPSS simulation, only major job points have been simulated. The following initialization and run times for the GPSS simulation were used:

N	Initialization Time (hours)	Run Time (hours)
10	6	2
20	12	4
30	18	6
*40	6	2

* Due to the lengthy computations involved, this sample is run for an indication only.

For N = 10, 20, and 30, a copy of the output is at Figures V-1, V-2, and V-3 respectively.

Comparison of
Basic Model and GPSS

	Q(T)		Tr(N)		%error
N	GPSS	BM	GPSS	BM	
10	7.77	7.69	4.32	4.50	4.2
20	14.19	13.25	6.13	7.64	24.6
30	18.73	16.23	9.02	12.73	41.1
*40	20.66	17.57	14.04	19.14	36.3

Table V-1

GPSS SIMULATION OF

BASIC MODEL, N = 20


```
NUMBER OF JOBS IS  20
INITIALIZATION HOURS ARE  12.00
DATA COLLECTION HOURS ARE   4.00
THINK TIMES ARE - A:  15000
                  B:  15000
                  C:  15000
                  D:  15000
SERVICE TIMES(MSEC) ARE -
                        FIP:  EXP   600
                        COMP: EXP  1375
                        DBM:  EXP  4125
                        PEP:  EXP  6600

THE JOB COUNT BY ROUTE
 FULLWORD MATRIX RTCNT

      ROW/COLUMN      1          2          3          4

       1          10455       1759        601        815
       2            767        129         44         59


COMPOSITE MEAN R TIME IS  6133.804
 TABLE  RESR1
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
           10455               1533.592                    1407.000
 TABLE  RESR2
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
            1759              17737.531                    8096.000
 TABLE  RESR3
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
             601              32938.945                   12832.000
 TABLE  RESR4
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
             815              20335.535                   10416.000


QUEUE STATISTICS
 QUEUE      AVERAGE      AVERAGE
          CONTENTS    TIME/TRANS
 TERMQ      14.194     14978.070
  FIPQ       1.806      1547.640
 COMPQ       1.442      2986.607
  DBMQ       1.422      8676.140
  PEPQ       1.134     11533.371


PROCESSOR STATISTICS
FACILITY    AVERAGE     NUMBER      AVERAGE
         UTILIZATION  ENTRIES    TIME/TRAN
    FIPF       .688      16808      589.957
   COMPF       .675       6953     1398.170
    DBMF       .693       2359     4230.437
    PEPF       .649       1816     6601.636


PROCESSOR ENT COUNT BY ROUTE
 FULLWORD MATRIX   ENT

      ROW/COLUMN      1          2          3          4          5

       1          10455       1759        601        815      13630
       2          10456       3519       1203       1630      16808
       3              0       3518       1802       1630       6950
       4              0       1760        601          0       1760
       5              0          0        600        816       1416
```

Figure V-2

GPSS SIMULATION OF

BASIC MODEL, N = 30

```
NUMBER OF JOBS IS  30
INITIALIZATION HOURS ARE  18.00
DATA COLLECTION HOURS ARE   6.00
THINK TIMES ARE - A:  15000
                  B:  15000
                  C:  15000
                  D:  15000
SERVICE TIMES(MSEC) ARE -
                        FIP:  EXP    600
                        COMP: EXP   1375
                        DBM:  EXP   4125
                        PEP:  EXP   6600

THE JOB COUNT BY ROUTE
 FULLWORD MATRIX RTCNT

      ROW/COLUMN      1          2          3          4

       1           21411       3070       1031       1462
       2             793        113         38         54


COMPOSITE MEAN R TIME IS  9021.246
 TABLE  RESR1
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
           21411               3159.893                    2496.000
 TABLE  RESR2
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
            3070              27224.417                   10048.000
 TABLE  RESR3
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
            1031              47819.867                   15104.000
 TABLE  RESR4
 ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION
            1462              29275.980                   13248.000


QUEUE STATISTICS
 QUEUE       AVERAGE       AVERAGE
           CONTENTS     TIME/TRANS
 TERMQ       18.730      14987.097
  FIPQ        4.846       3217.169
 COMPQ        2.534       4500.992
  DBMQ        2.247      11825.261
  PEPQ        1.641      14190.285


PROCESSOR STATISTICS
FACILITY   AVERAGE      NUMBER      AVERAGE
         UTILIZATION  ENTRIES     TIME/TRAN
    FIPF      .898       32538       596.482
   COMPF      .791       12160      1388.582
    DBMF      .787        4106      4143.199
    PEPF      .726        2494      6290.730


PROCESSOR ENT COUNT BY ROUTE
 FULLWORD MATRIX   ENT

     ROW/COLUMN      1          2          3          4          5

       1           21411       3070       1031       1462      26974
       2           21410       6139       2062       2925      32536
       3               0       6142       3095       2925      12162
       4               0       3070       1031          0       3070
       5               0          0       1033       1463       2496
```

Figure V-3

From the results in Table V-1, it is apparent that the Basic Model may not predict response times as accurately as one would desire. This is true even when one considers that for EQN modelling, we should be satisfied with response time accuracies with a tolerance of up to 30% [Buzen and Denning 77]. Small values of N are producing good results and tend toward the unacceptable range near N = 20. The figures do, however, provide insights as anticipated. Recall that the design criteria for the balanced system was with a saturation of N* = 24. In other words, as long as we are below saturation, we obtain usable results but, above saturation, the assumptions made in the development of the Basic Model show their effect dramatically. The principal assumption made was in the assimilation of the k job routes into one in order to utilize a quick and rapid computation facility such as the single job type ASQ. As explained in Chapter IV, we have assumed a job load distribution in the CPN as (.6, .2, .1, .1). In fact, this is not true as can be verified by the frequency distribution in the job count by route matrix of the GPSS output which shows the following load distributions:

N	Job load Distributions
10	(.729, .150, .053, .067)
20	(.767, .129, .044, .059)
30	(.793, .113, .038, .054)

Due to their faster response, we expect to find a large number

of route 1 jobs and a fewer number of jobs of routes 2, 3, and 4 in the CPN. Since route 1 jobs require only the FIP processing function, whereas the others require two units of processing from the FIP and at least two other processing functions, it is natural to find this skewed job load distribution from that assumed. Furthermore, to a large degree, this frequency distribution will depend on the network structure and the processing functions.

The utilization of the processors remains relatively balanced as intended throughout the range of N as shown in Table V-2.

GPSS Utilization Summary
of a Balanced System

N	BM (all $i \neq T$)	GPSS			
		FIP	COMP	DBM	PEP
10	.385	.394	.437	.425	.412
20	.663	.688	.675	.693	.649
30	.811	.898	.781	.787	.726
40	.879	.983	.813	.813	.804

Table V-2

This is expected since it is realized that utilization figures are affected less by the job load distribution than are response times [Buzen and Denning 77]. Again the effect of the loading of the FIP by route 1 jobs is shown by a utilization slightly greater than the balanced values of the Basic Model.

Summary and Motivation for Further Refinement

The Basic Model (BM) is shown to provide good results when used with a small number of jobs, that is, within the saturation design criteria. However, near the upper limit of the saturation the response time error is greater than desired. As is expected, the utilizations provide much better results throughout the range. A significant advantage of this model is its simplicity of use when a general purpose program such as ASQ is available. Both trade-off analysis and balanced design may be accomplished very quickly. Consequently, it is expected that the Basic Model may be effectively utilized to give insights but we are motivated to search for yet a more accurate model, realizing that a trade-off in terms of simplicity both in use and in computation may be required. We should also note that in the Basic Model the service rates and service disciplines at the functionalized processors must be identical for all routes. This restriction is not felt to be significant since a particular route may require multiple visits at a functionalized processor. The multiplicity of visits may account for differing processing requirements by route.

Chapter VI: FOUR JOB TYPES MODEL USING INTEGER ITERATION

General Approach

The greatest disadvantage of the Basic Model presented in Chapter IV is the assumption required to assimilate the k job types (routes) into one. This assimilation was done to simplify the computation procedure and to take advantage of the Duke ASQ general purpose program. Another approach would be to use one of the general purpose programs with multi-job capabilities such as the ASQ from Information Research Associates [1975]. This possibility was investigated and some trial runs were made using a version available at Federal Simulations in the Department of Defense executing on a Honeywell (GE) 635. However, due to the constraints of providing only three job types and a total of 20 jobs, this program was not felt to be entirely satisfactory for our purposes. Furthermore, the amount of CPU execute time required was relatively high, making the program expensive to use. Since the exact algorithms used for solution were not immediately available, it is assumed that additional time was required because of the general purpose nature of the program. The other simulation program, QNET4, has not been released for use outside IBM. It too, however, restricts the user to three job types instead of our desired four.

Consequently, another approach, which is pursued in this chapter, is to develop a special purpose program employing the solution forms of Buzen [1973] and Williams and Bhandiwad [1976]. Not only are we able to tailor the program to model our computer network, but we can ensure that some of the most computationally efficient techniques and procedures are employed.

Solution Forms with Normalization Constant

The general solution form for a single job type in a closed FQN model ([Gordon and Newell 67], [Williams and Bhandiwad 76]) is based on the state of the network at any time, denoted by a job vector, $\underline{n} = [N(1), N(2), \ldots, N(n)]$, where N(i) is the number of jobs at node i (both those in service and those waiting in the queue). Since each job is either in service or waiting in one of the queues, we must have

$$\sum_i N(i) = N \qquad \text{(VI-1)}$$

The solution form for the probability of being in a particular state, often referred to as the product form solution, is

$$Pr[N(1), N(2), \ldots, N(|M|)] = (1 / G(N)) \prod_i X(i) ** [N(i)] \qquad \text{(VI-2)}$$

where

$$X(i) ::= v(i) / u(i) \qquad \text{(VI-3)}$$

Note that the v(i) represents the relative visits to node i as solved by the balance equations (IV-40). Consequently the X's are often referred to as the relative utilizations. The normalizing constant, G, is determined by the requirement that the probabilities summed over all states is unity. Therefore

$$G(N) = \sum_{St(N,|M|)} \prod_i [X(i) ** (N(i))] \qquad \text{(VI-4)}$$

where the sum is over all $\underline{n}$ in the state space of N jobs and |M| nodes, denoted as St(N,|M|). It is then shown that the utilization of the i-th node with N jobs in the network is

$$U(i,N) = Pr[N(i) \geq 1] = X(i) * G(N-1) / G(N) \qquad (VI-5)$$

From the utilization, the thruput may be computed as follows.

$$T(i,N) = U(i,N) * u(i) \qquad (VI-6a)$$

$$= [X(i) / z(i)] * G(n-1) / G(N) \qquad (VI-7)$$

where z(i) is the mean service time at node i (equal by definition to 1 / u(i)).

Solution Forms for Multiple Job Types

At this juncture we should recall that the solution forms in the previous paragraph are for the situation of only one job type. Basket et al [1975] have extended the work of Jackson [1957] and Gordon and Newell [1967] to provide solution forms for multiple job types. Buzen [1971, 1973] developed the idea of using the normalizing constant to characterize network properties. He also provided the recursive specification of the normalizing constant as a computational technique. Subsequently, Williams and Bhandiwad [1976] neatly present the solution forms for two job types requiring an extension to k job types in our case. We can designate Nk(n) as the number of jobs of the k-th route or job type in the complete network n such that the sum over k of Nk(n) equals N, the total number of jobs in the network. Likewise, Nk(i) is the number of jobs of the k-th route at the i-th node in the network. Then for two job types or routes, k = 1, 2, the solution form is

$$\Pr[N1(1),N2(1);\ldots;N1(i),N2(i);\ldots;N1(|M|),N2(|M|)] = [1 / G2] \prod_i PROD2(i) \qquad \text{(VI-8a)}$$

where

$$PROD2(i) ::= ([N1(i) + N2(i)]! / [N1(i)!\ N2(i)!]) * X1(i)^{N1(i)} * X2(i)^{N2(i)} \qquad \text{(VI-8b)}$$

where G2 is again the normalizing constant but in this case for

two job types. Thus

$$G2 = \sum_{St[N1(n),N2(n),|M|]} \prod_{i} PROD2(i) \qquad \text{(VI-9)}$$

for all [N1(i),N2(i)] such that

$$\sum_{i} Nk(i) = Nk(n) \text{ for } k = 1, 2.$$

Extending equation VI-8 to k job types or specifically in our case where k=4 we have

$$Pr[N1(1),\ldots,Nk(1);\ldots;N1(i),\ldots,Nk(i)] = [1 / Gk] \prod_{i} PRODk(i) \qquad \text{(VI-10a)}$$

where

$$Prodk(i) = \frac{[\sum_{k} Nk(i)]!}{\prod_{k}[Nk(i)!]} * \prod_{k} Xk(i) ** [Nk(i)] \qquad \text{(VI-10b)}$$

and again G(k) is the normalizing constant for k job types:

$$Gk = \sum_{St[N1,N2,\ldots,Nk,|M|]} \prod_{i} PRODk(i) \qquad \text{(VI-11)}$$

For the sake of simplicity and clarity in the program implementation and in further notation, the k superscript, indicative of the job type or route, will be specifically designated by the job types as A, B, C, and D. Likewise the corresponding relative utilizations for each job type at the i-

th node can be respectively designated as W(i), X(i), Y(i) and Z(i) where

X1(i) becomes W(i)

X2(i) becomes X(i)

X3(i) becomes Y(i)

X4(i) becomes Z(i)

Other variables will have the letter A, B, C, or D appended in lieu of the superscript appendage k. For example:

v1(i) becomes vA(i)

z2(i) becomes zB(i)

N3 becomes NC

T4(i) becomes TD(i)

and so forth.

Recall that the vk(i) is the relative visits to node i for job type (route) k found by the balance equations (IV-40) and the zk(i) = 1 / u(i). In accordance with the previous specification of Chapter III, the s(i) = <u(i), a(i)> are identical for all routes. Strictly speaking, we have the capability to specify s(i) for each route. Following the previous nomenclature, we would have sk(i) = <uk(i), ak(i)>.

Now for our specific case the solution form becomes

$$Pr[NA(1),NB(1),NC(1),ND(1);\ldots;NA(4),NB(4),NC(4),ND(4)] = [1 / G(NA,NB,NC,ND)] \prod_i Prod4(i) \qquad (VI-12a)$$

where

$$Prod4(i) = [NA(i) + NB(i) + NC(i) + ND(i)]! / [NA(i)! NB(i)! NC(i)! ND(i)!] * W(i)^{NA(i)} * X(i)^{NB(i)} * Y(i)^{NC(i)} * Z(i)^{ND(i)} \quad \text{(VI-12b)}$$

Looking at four job types, the utilization at node i is

$$U(i,NA,NB,NC,ND) = [W(i)\,G(NA-1,NB,NC,ND) + X(i)\,G(NA,NB-1,NC,ND) + Y(i)\,G(NA,NB,NC-1,ND) + Z(i)\,G(NA,NB,NC,ND-1)] / G(NA,NB,NC,ND) \quad \text{(VI-13)}$$

Similarly to equation VI-6, the thruput for each job type is

$$TA(i) = U(i,NA,NB,NC,ND) * u(i) \quad \text{(VI-14a)}$$

$$= [X(i) / zA(i)]\, G(NA-1,NB,NC,ND) / G(NA,NB,NC,ND) \quad \text{(VI-14b)}$$

This equation holds similarly for job types B, C, and D.

Computation of the Normalization Constant

From the previous sections we have seen that the normalization constant, G, is central to solutions for utilization and thruput. Therefore, we need to examine the computational methods available for G. One of the more efficient methods in the single job type case is presented by Buzen [1973] and then by Williams and Bhandiwad [1976].

$$g(nA, m) = g(nA, m-1) + W(m)\, g(nA-1, m) \qquad \text{(VI-15)}$$

The latter reference extends the single job type algorithm to multiple job classes. In the case of two job types, it is

$$\begin{aligned} g(nA, nB, m) = {} & g(nA, nB, m-1) \\ & + W(m)\, g(nA-1, nB, m) \\ & + X(m)\, g(nA, nB-1, m) \end{aligned} \qquad \text{(VI-16)}$$

Extended to four job types, it is

$$\begin{aligned} g(nA, nB, nC, nD, m) = {} & g(nA, nB, nC, nD, m-1) \\ & + W(m)\, g(nA-1, nB, nC, nD, m) \\ & + X(m)\, g(nA, nB-1, nC, nD, m) \\ & + Y(m)\, g(nA, nB, nC-1, nD, m) \\ & + Z(m)\, g(nA, nB, nC, nD-1, m) \end{aligned} \qquad \text{(VI-17)}$$

The m-th subscript indicates the m-th node in the network and in our case has a maximum value of $|M|$. The nk is such that $1 \leq nk \leq Nk$. The final value of g, G, is obtained when m reaches its maximum value.

It is seen from the above equations that this is a recursive specification such that the current value of g is

based on previously computed values. It is a matter of establishing the initial values so that the basic equation of the solution form is met. For the single job type, $g(0,m) = 1 \; \forall \; m$ and $g(nA,0) = 0 \; \forall \; nA$ are the initialization requirements. Furthermore, it is noted that although g is in the form of a two-dimensional matrix, only the current column vector need be retained for computation and not the entire matrix. Consequently, computational storage requirements are reduced considerably.

In analyzing the equations VI-16 and VI-17 as extensions of equation VI-15, we note that the minimum dimensions required for computation of G are NA, NB, NC, and ND. The m value for the nodes is maintained only as the current computation of g. In order to visualize more clearly how the computational procedure is extended to four or more job types, Figure VI-1 illustrates the case of two job types. Here we have let a and b be the running variables for NA and NB respectively in lieu of the nA and nB. In this case the full g would be a three-dimensional matrix or cube and the initial values would lie adjacent to the two faces of the cube as shown. These initial values represent the case where one of the job types has a zero value, e.g. g(0,b,m) or g(b,m) and g(a,0,m) or g(a,m). But to obtain the g(a,m) and g(b,m) we need to employ the procedure for the single job type case.

Schema for G Computation

for Two Job Types

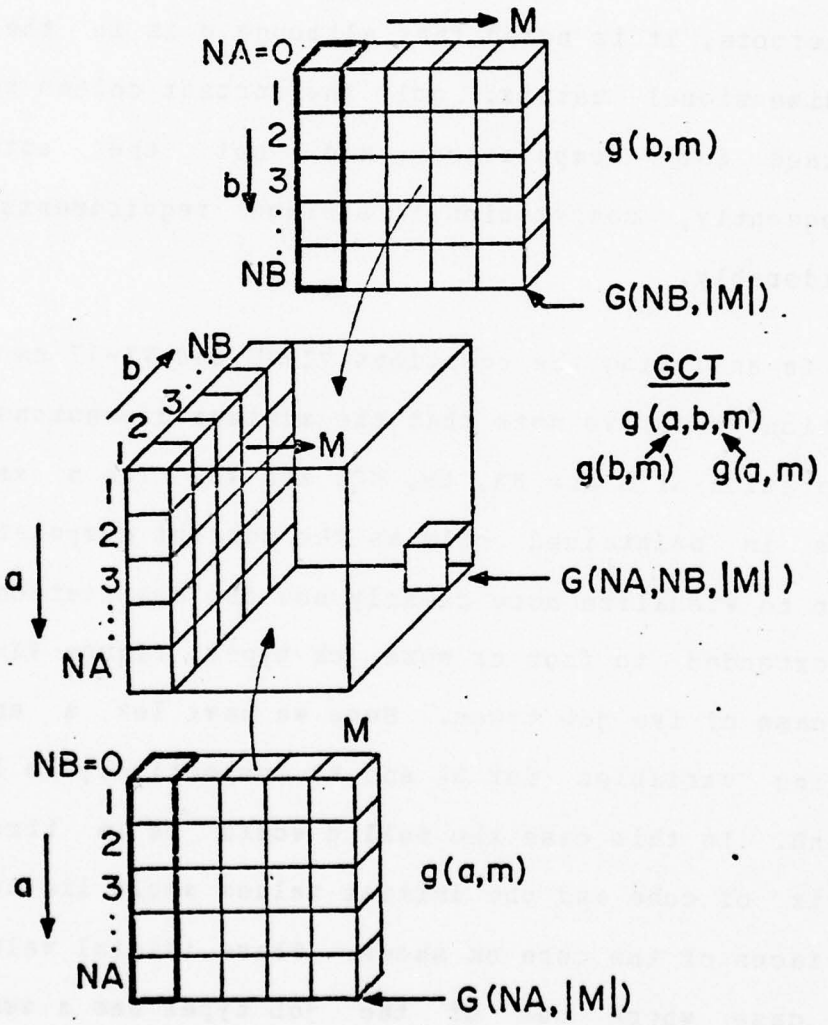


Figure VI-1

As explained in the single job type case, it is not necessary to compute the entire two-dimensional matrix represented in Figure VI-1 as a face to the cube but one column vector for each m value would suffice. Likewise only one "m slice" of the cube is required for the current computation of g. In order to simplify the computational algorithm, however, we will compute the entire initial value matrices although the m dimension of the final g computation will be eliminated.

Using the procedure just described, a G Computation Tree (GCT) is developed:

Computation Tree for Two Job Types

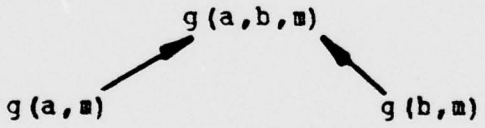


Figure VI-2

For more than two jobs, hypercubes and hyperplanes result. A GCT is shown at Figure VI-3 for three jobs and at Figure VI-4 for four jobs, our case of interest. The following correspondence of running variables is used throughout:

nA becomes a

nB becomes b

nC becomes c

nD becomes d

This computational procedure is implemented by the program listed at Appendix C.

G Computation Tree

for Three Job Types

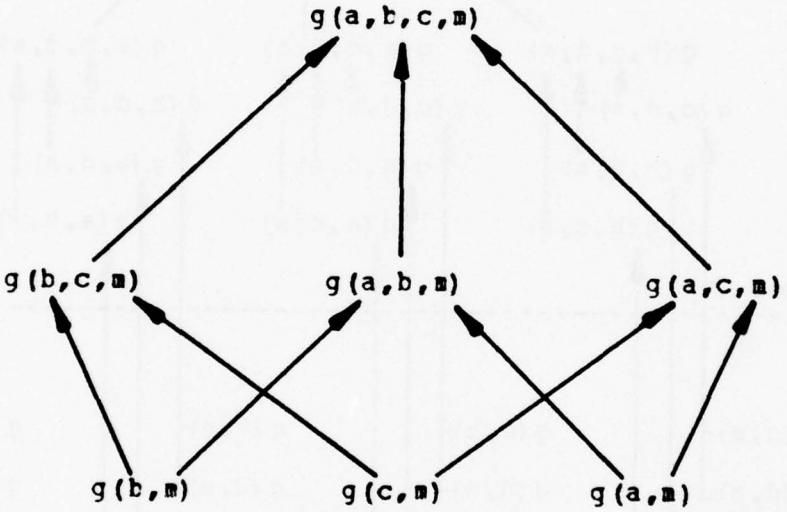


Figure VI-3

G Computation Tree

for Four Job Types

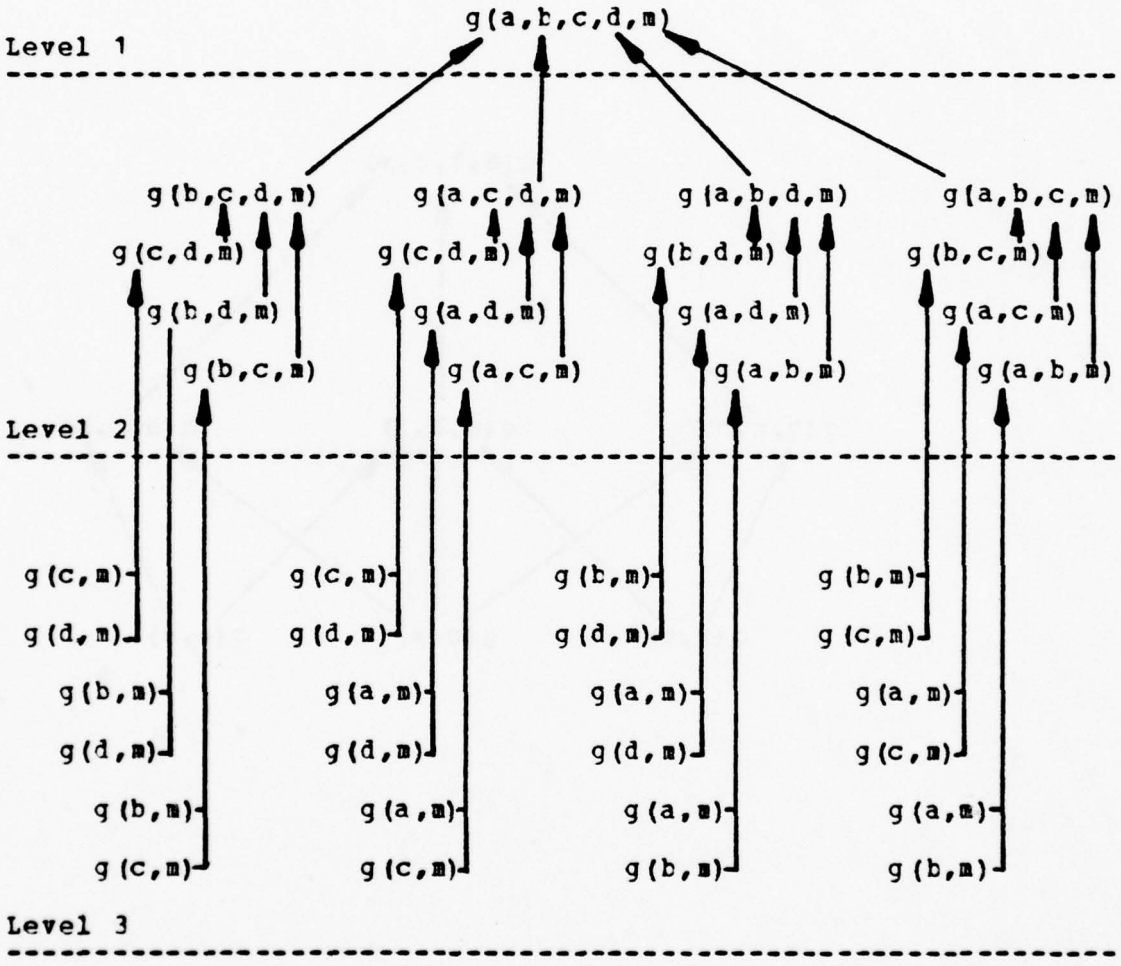


Figure VI-4

In order to provide additional assurance and a greater degree of verification, several test cases of the G Normalizatien Computation were considered.

1. Single job type computations were ccmpared against Buzen's example [1973] and against the Duke ASQ employing the five network modes as described in Chapter III.
2. Two job type computations were compared against a small two-node network computed by hand and also run using the Information Research Asscciates [1975] ASQ at Federal Simulations.
3. Three job type computations were run at Federal Simulations within the constraints of the ASQ implemented there. The model run was similar to that of Chapter III. Only r(4) was eliminated and q(3) was increased from 0.1 to 0.2.

All of the above cases yielded exact comparisons for the utilization and thruput. It should be noted that a direct comparison of G values was not made since they are a function of the relative utilizations which in turn are dependent on the solution to the balance equations (IV-40) which dc not have a unique solution.

Model Formulation and Implementation

By using equations VI-13 and VI-14 and the computation of the normalization constant, we can compute the utilization and the thruput at a particular node. What we are interested in obtaining, however, is the response time as a function of the number of users Tr(N). Recall from Figure VI-5 (same as Figure IV-1) that the response time is the time a job spends in the CPN.

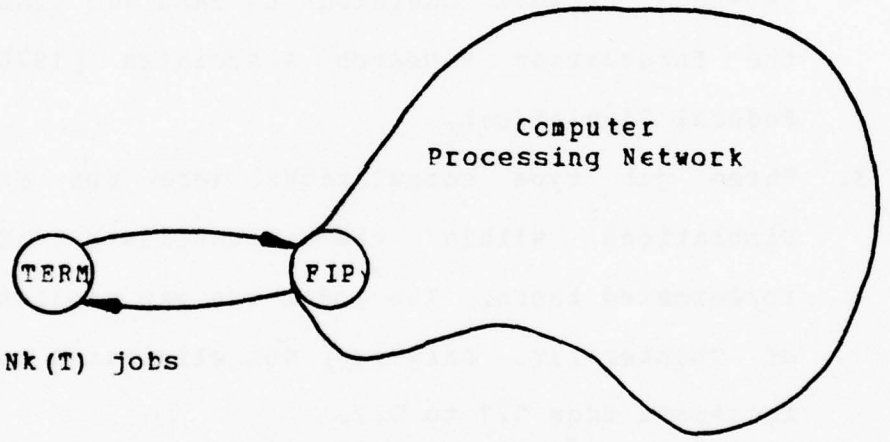


Figure VI-5

It is recognized that we can apply Norton's equivalent [Chandy, Herzog and Woo, 75a and 75b] to the Computer Processing Network. The CPN is then replaced by its short-circuited thruput equivalent. The short circuit thruput for each job type is computed directly from equation VI-14 as long as the $v_k(i)$ $\forall\, i > 1\ \forall\, k$ are computed based on $v_k(1) = 1\ \forall\, k$. Now we can apply a job load vector $\underline{n}$ = (NA,NB,NC,ND) to the CPN and

determine the equivalent short-circuit thruput. In other words, with the CPN loaded by n, the equivalent thruput of the CPN for each job route, Tk(CPN), is obtained. In fact, we may compute Tk(CPN) as a function of the number of each job type in the CPN (Nk(CPN)) varied over its range of values to form distinct job vectors for Tk(CPN). As a result, there is a Tk(CPN) for each value of Nk(CPN).

We now further recognize that we can obtain a set of thruputs by varying each job type over the range of its values in the CPN while holding the number of other job types fixed at the maximum values of the job load vector. This will yield a minimum thruput for each value of each job type since the CPN is maximally loaded with the remaining job types. As a consequence, we will obtain an upper bound on response time. To obtain this response time we have used the Duke ASQ for the solutions as a matter of convenience. We represent the equivalent network as two nodes, the TERM and the CPN, for each of the k job types.

Figure VI-6

The TERM is as represented previously in Chapter III and by equation III-2 as u(T) = N / Z where N is now Nk and Z would

be zk. The effective service rates of the CPN is the set of thruputs obtained for each job type such that

$$uk(CPN) = \{Tk(nk)\} \quad 1 \leq nk \leq Nk \qquad (VI-18)$$

(The Tk is for the CPN node of Figure VI-6 so the CPN subscript is implied for the remainder of this chapter.) From ASQ we solve for the number of jobs of the respective type in the CPN (the queue length of the CPN node) and the response time. The number of jobs in the CPN is a new approximation to the job load for the respective job type. Doing this for the k job types, we now have a new job load vector $\underline{n}'$. With this new job load vector we can again compute the thruputs and obtain a new loading of the CPN. We are restricted to integer values, however, since the computational procedure for G is restricted to integer job values. We will converge to an approximate value as we obtain better estimates of the integer values for the job load vector.

We are aware that there will be an exact solution for the job load vector if we were to accommodate other than integer values [Baskett et al 75]. Starting with an assumption of maximum load, we would obtain successively better approximations until the exact value is reached. In our case we will obtain the best integer load vector near the exact values.

The algorithm for the Iterative Model (IM) is formulated as follows:

1. Determine the relative utilizations Xk(i) based on vk(T) = 1 ∀ k.
2. Assume a maximum job load vector $\underline{n}$ = (N1,N2,...,Nk).
3. Solve for the CPN thruput vector for each job type [Tk(nk)], assuming that the other job types maximally load the CPN. (The program at Appendix III is used.)
4. Determine a new job load value by using ASQ as in Figure VI-6 and the uk(CPN) as in equation VI-18.
5. Approximate each of the job load values by the nearest positive integer.
6. Compare previous job load values.
7. If there is no change, then the best integer load value has been determined; else set a new load vector and repeat starting at step 3.
8. Determine the [Trk(Nk)] from the TERM queue length and equation IV-10.

A composite response time must be formulated based on the k response times. This is done by weighting the response times by the job load distribution or the proportion of the k job types flowing through the CPN which is based on the ratio of job type thruputs.

$$CTr(N1,N2,...,Nk) = Tr(N) = \sum_{k} p(k)\ Trk(Nk) \qquad (VI\text{-}19)$$

where

$$p(k) = \frac{Tk(Nk)}{\sum_{k} Tk(Nk)} \quad \forall\ k \qquad \text{(VI-20)}$$

Once the G values have been computed for the value of |M| nodes, the data may be retained and used to provide G values for any job load vector where the individual job type values are less than or equal to those initially computed for G. Consequently, for the second and subsequent passes of step 3, the G values need not be recomputed. An interactive program (TPUT) at Appendix D provides such a capability. Also, if memory size constraints are not inhibiting, then it is possible to compute G values for the maximum anticipated load vector and retain these values in a data file, thus eliminating the need for further G computations.

Verification Using GPSS

The same GPSS results as presented in Chapter V were compared against our integer iterative model. The Network Description found in Chapter III and used for the Basic Model and the balanced service rates of Chapter IV are used in this verification.

Computation for NA=6, NB=2, NC=ND=1, and N=10

Iteration Number	Computed Thruput [Tk(nk)]	Proportion of Types p(k)	Tr[N(k)]	Tr(N)
	1.318	0.883	1.08	--
1	0.099	0.066	19.1	
	0.031	0.021	32.3	
(6,2,1,1)	0.044	0.030	22.7	
Computed job load vector =		[0.402	1.121	0.683 0.602]
	1.290	0.873	1.00	3.22
2	0.086	0.058	16.7	
	0.039	0.027	25.6	
(1,1,1,1)	0.063	0.043	15.9	
Computed job load vector =		[0.377	1.054	0.631 0.514]

Table VI-1

Computation for NA=12, NB=4, NC=ND=2, and N=20

Iteration Number	Computed Thruput [Tk(nk)]	Proportion of Types p(k)	Tr[N(k)]	Tr(N)
	1.243	0.855	1.91	--
1	0.117	0.081	37.1	
	0.040	0.028	48.9	
(12,4,2,2)	0.054	0.037	35.9	
Computed job load vector =	[1.357	2.715	1.530	1.410]
	1.165	0.823	1.79	--
2	0.147	0.104	19.4	
	0.053	0.038	36.0	
(1,3,2,1)	0.051	0.036	30.7	
Computed job load vector =	[1.282	2.253	1.142	1.348]
	1.210	0.841	1.68	5.44
3	0.133	0.093	19.7	
	0.035	0.024	47.3	
(1,2,1,1)	0.060	0.042	25.4	
Computed job load vector =	[1.206	2.270	1.519	1.258]
	1.206	0.839	1.69	5.73
4	0.120	0.084	22.5	
	0.059	0.041	32.1	
(1,2,2,1)	0.051	0.036	30.4	
Computed job load vector =	[1.213	2.400	1.363	1.344]
		0.840	1.68	5.58
Final result		0.088	21.1	
		0.032	39.7	
		0.039	28.1	

Table VI-2

In Table VI-2 the exact value for convergence lies somewhere between the values obtained in iterations three and four. Consequently, there is an oscillation between the two sets of values. This situation is due to the restriction of using only integer values for the job load vector. A judicious analysis is made to obtain:

$$\underline{p} = [.840 \quad .088 \quad .032 \quad .039]$$

$$\underline{Tr} = [1.68 \quad 21.1 \quad 39.7 \quad 28.1]$$

and

$$Tr(N) = 5.58 \text{ seconds}$$

as the final results for twenty jobs.

Computation for NA=18, NB=6, NC=ND=3, and N=30

Iteration Number	Computed Thruput [Tk(nk)]	Proportion of Types p(k)	Tr[N(k)]	Tr(N)
	1.206	0.724	3.78	--
1	0.126	0.151	44.4	
	0.046	0.055	64.1	
(18,6,3,3)	0.059	0.070	49.4	
Computed job load vector =	[3.626	4.484	2.431	2.301]
	1.118	0.803	3.61	--
2	0.166	0.119	27.4	
	0.040	0.029	64.4	
(4,5,2,2)	0.068	0.049	35.0	
Computed job load vector =	[3.494	3.876	2.434	2.101]
	1.128	0.807	3.51	9.43
3	0.154	0.110	27.8	
	0.043	0.031	59.4	
(3,4,2,2)	0.042	0.052	32.7	
Computed job load vector =	[3.413	3.895	2.395	2.057]

Table VI-3

The comparison of this model and GPSS can be made not only on the final composite response time, Tr(N), but also with the individual job type or route response times, Trk[nk]. The proportionality of job types in the CPN for this model is also available and is compared to that computed by GPSS. Comparisons of these values yield further insight into the interaction or interference of the job types within the computer network.

Comparison of
Iterative Model and GPSS

N	p		Tr		Tr(N)	
	Iter	GPSS	Iter	GPSS	Iter	GPSS
10	.873	.729	1.00	.891	3.22	4.32
	.058	.150	16.7	10.7		
	.027	.053	25.6	21.3		
	.043	.067	15.9	13.7		
			error in Tr(N) = 25 %			
20	.840	.767	1.68	1.53	5.58	6.13
	.088	.129	21.1	17.7		
	.032	.044	39.7	32.9		
	.039	.059	28.1	20.3		
			error in Tr(N) = 9.0 %			
30	.807	.793	3.51	3.16	9.43	9.02
	.110	.113	27.8	27.2		
	.031	.038	59.4	47.8		
	.052	.054	32.7	29.3		
			error in Tr(N) = 4.5 %			

Table VI-4

A review of the percent error at each of the comparisons reveals two major items. First, the model provides quite comparable results to the Basic Model with sufficient accuracy to offer somewhat better overall insights (see Table V-1). Secondly, as the total number of jobs is increased, the Iterative Model yields increasingly better results. This situation is the reverse of that given by the Basic Model. The fact that larger values of N lead to better results is expected. The error introduced due to integer representation of the job load vector becomes less pronounced as N is increased.

Summary and Motivation for Further Refinement

The four job type model using integer iteration (IM) provides a good model and meets our objective of obtaining insights concerning response times, the major performance criterion. In particular, it complements the Basic Model by providing greater accuracies with large N. However, we do incur a greater complexity of computation.

A useful result of this model formulation has been the development of the specialized programs for the multijob type normalization constant computation based on the GCT (Appendix C) and the solution of the short circuit thruput values (Appendix D) using the Norton equivalent. Both of these programs have been developed to operate in an interactive mode using the IBM TSO. Although the Duke ASQ is used for the evaluations in step 4 of the algorithm, we have the basis for our own program solution since the computation of G is readily available.

Although we have obtained a more accurate model for larger values of N, we are motivated to develop a still more accurate model throughout the entire range of N. Based on the knowledge gained by development of the Iterative Model, it appears that such a new model may be at hand.

Chapter VII: EXACT HNCFM MODEL

General Approach

In the formulation of the Iterative Model (Chapter VI) we have reduced our network to a two node representation of the TERM and CPN, Figure VII-1 (same as Figure VI-6).

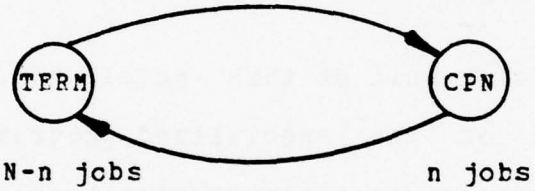


Figure VII-1

We have regarded the two node representation as a closed system of two load-dependent devices. For the moment, let us suppose that we are dealing with one job class and a total of N jobs and let n be the number of jobs in the CPN. For this specific case we find that Buzen and Denning [1977] have provided a means of obtaining exact solutions. The solutions provided yield a direct computation of thruput as a function of N, T(N) and queue lengths. In our specific case, the queue length we are interested in is the average number of jobs, n, in the CPN.

$$T(N) = [(N / Z)\, h(N-1, m)] / h(N, m) \qquad \text{(VII-1)}$$

$$n = N [1 - (h(N-1, m)) / h(N,m)] \qquad \text{(VII-2)}$$

All parameters are as previously defined except for the h, which is the value obtained in the computation of a special normalizing constant, H, which in turn is related to our familiar G normalizing constant of Chapter VI. Being able to compute h permits us to solve equations VII-1 and VII-2. Furthermore, we recognize that the mean response time (time spent in the CPN) can then be simply computed [Buzen 76a] as:

$$Tr(N) = n / T \qquad \text{(VII-3)}$$

The H relationship is established as

$$H = \sum_{n=0}^{N} \frac{N!}{(N-n)!\ Z^{**}n}\ G(n) \qquad \text{(VII-4)}$$

which leads to an iterative solution technique analogous to the G computation. The recursive specification for h is

$$h(a, m) = h(a, m-1) + a / Z\ W(m)\ h(a-1, m) \qquad \text{(VII-5a)}$$

$$\text{for } a > 0 \text{ and } m > 0$$

with the following initial conditions:

$$h(0, m) = 1 \qquad \text{for } m \geq 0 \qquad \text{(VII-5b)}$$

$$h(a, 0) = 1 \qquad \text{for } a \geq 0 \qquad \text{(VII-5c)}$$

Here, the a is a running variable representing the number of the maximum N jobs in the two node network representation of Figure VII-1. We immediately recognize that the recursive specification of VII-5 is the same as that for g, equation

VI-15, with the exception of the multiplicative term, a / Z, for the relative utilization, W(m). It is this multiplicative term which acccunts for the think time of jobs at TERM.

We have already established a means of computation for the G normalization constant in Chapter IV and can simply use the same technique for the solution of H. In fact, if we take an engineering approach tc the program, we can by direct analogy obtain the solutien form of H for multiple job classes. We have already noted that the sole difference between g and h is the multiplicatien of the relative utilizatien by the term a / Z. Equations VI-16 and VI-17 show the extensien of g from the single job type case to that for twc and four job types, respectively. Recall that a is the running variable for the number of type A jobs in the network and zA is the think time of job type A. Then the multiplicative term in the H normalization computation for four job types is:

W(m) becomes a / zA * W(m)

X(m) becomes b / zB * X(m)

Y(m) becomes c / zC * Y(m)

Z(m) becomes d / zD * Z(m)

Note that we have the ability to consider individual think times for each job route, not possible with the Basic Model. Thus h for four job types is

$$h(nA, nB, nC, nD, m) = h(nA, nB, nC, nD, m-1) + a / zA * W(m)\ h(nA-1, nB, nC, nD, m) + b / zB * X(m)\ h(nA, nB-1, nC, nC, m) + c / zC * Y(m)\ h(nA, nB, nC-1, nD, m) + d / zD * Z(m)\ h(nA, nB, nC, nD-1, m)$$

(VII-6)

Model Formulation and Implementation

Due to our previous work with the iterative model of Chapter VI, the HNORM Model formulation and implementation is a natural result. We have already established the technique of iterative computation of g for multiple job classes which was shown as a GCT. The same computational tree structure for the recursive specification of h is also applicable here. Now, having h for multiple job classes, we can extend the equations for thruput, average number of jobs in the CPN, and mean response time for multiple job types to obtain

$$T(NA) = [NA / zA\ H(NA-1, NB, NC, ND)] / H(NA, NB, NC, ND) \quad \text{(VII-7)}$$

$$nA = NA\ [1 - H(NA-1, NB, NC, ND)] / H(NA, NB, NC, ND) \quad \text{(VII-8)}$$

$$Tr(NA) = nA / T(NA) \quad \text{(VII-9)}$$

and so forth for job types B, C, and D.

We establish the two node representation of the network as in Figure VII-1 and then solve for the parameters of interest. We may additionally obtain the proportion of jobs of a particular type, p(k), that are in the CPN by taking the proportion of thruputs as was done in equation VI-20:

$$p(k) = \frac{Tk(Nk)}{\sum_{k} Tk(Nk)} \quad \forall\ k \quad \text{(VII-10)}$$

As was done in the previous model (IM), the results obtained using the H normalization procedure were verified against test runs on a more limited basis.

1. The single job type example provided by Buzen and Denning [1977].
2. A simple two node, two job type case compiled by hand.
3. A three job type computation run at Federal Simulations where the model of Chapter III was modified by the elimination of r(4) and q(3) was increased from 0.1 to 0.2.

The results obtained were in all cases exactly the same using this model with the H normalization.

For the network of functionalized processors using the balanced service rates as before, the Exact HNORM Model gave the following results:

Exact H Model Results

N	p	Tr	Tr(N)
10	.731	.907	4.38
	.148	11.1	
	.053	21.6	
	.068	13.6	
20	.766	1.60	6.20
	.130	17.6	
	.044	33.2	
	.060	20.6	
30	.793	3.22	9.08
	.115	26.8	
	.038	47.9	
	.053	30.1	

Figure VII-2

Although only the job points of 10, 20, and 30 are shown, all possible values for job types may be readily computed since all H values are ascertained. In Appendix E is a program to compute the H values and the parameter values of interest (thruput, number of jobs in the CPN, and the mean response times). These latter parameter values are for NA, NB, NC, and ND. The values of interest may be readily obtained for any other network load (nA, nB, nC, nD) where $nk \leq Nk \ \forall \ k$ once the H values are obtained. In Appendix F there is an interactive program using the IBM TSO which takes the H values as a file input from the Exact HNORM Model Computation and computes the values of interest for any network load as specified above.

Verification Using GPSS

Following the procedure previously established, we have compared the HNORM Model (HM) results to the GPSS simulation. At Table VII-2 are the comparative results. It is evident that this model representation is, in fact, an exact computation when we acknowledge that the event driven simulation model will yield accuracies of 5-10 percent of the actual system.

Comparison of
Exact H Model and GPSS

N	p		Tr		Tr(N)	
	HM	GPSS	HM	GPSS	HM	GPSS
10	.731	.729	.907	.891	4.38	4.32
	.148	.150	11.1	10.7		
	.053	.053	21.6	21.3		
	.068	.067	13.6	13.7		
			error in Tr(N) = 4.0 %			
20	.766	.767	1.60	1.53	6.20	6.13
	.130	.129	17.6	17.7		
	.044	.044	33.2	32.9		
	.060	.059	20.6	20.3		
			error in Tr(N) < 1.0 %			
30	.793	.793	3.22	3.16	9.08	9.02
	.115	.113	26.8	27.2		
	.038	.038	47.9	47.8		
	.053	.054	30.1	29.3		
			error in Tr(N) < 1.0 %			

Table VII-2

Summary of Exact HNORM Model

We have called this the Exact HNORM Model because within the assumptions of EQN modelling we have obtained exact solutions for the parameters using the H normalization constant. This model can be easily used and provides the performance criteria in which we are interested. In fact it provides more flexibility than we have had before. Not only do we obtain individual route thruputs and response times, but we can easily designate individual route think times. This capability adds another degree of flexibility to our network modelling.

Use of the programs developed at Appendices E and F provide a means of ready model solution and network analysis.

Chapter VIII: CONCLUSIONS

General Conclusions

There are several conclusions to be drawn from this research concerning the development of the three EQN models presented and from the models themselves. First, recall that the basic objective was to develop a model that could be used for trade-off analysis of large scale computer networks and not just "local" computer systems. Secondly, we wanted a design aid to guide in the solution of service rates for the processors in the network. It was felt that these objectives could be obtained as long as we were willing to accept insights and not concentrate on details, select response time as a major performance criterion, and constrain the network model in terms of functionalized processors. Furthermore, by the use of functionalization, one is able to impart generality to the model in the sense that the network may be viewed as being composed of several defined functions. Since the model is constructed of functionalized processors, it is then the responsibility of the designer/analyst to ensure that the characterization of functions in the computer network is an accurate network representation.

Through the use of the exponential queuing network approach, we have seen three methods of solving the problem.

These methods have been presented to accommodate a large number of jobs and nodes so that an effective, total computer network may be modelled.

In the development of the models we have seen that the General Purpose EQN solution programs such as the versions of ASQ and QNET4 can work well with small networks, few job classes and few total jobs. They seem to be best suited for the network representation of a local computer system. When we wish to model on a larger scale it is necessary to develop more specific program models. Our approach through the use of the computer program modules of Appendices A - F permits the representation and solution of such models. In fact, with modification and extension of these modules, a general purpose system could be developed. Furthermore, the modules provide the job route (type) response times of the interactive system being modelled. Of the other known solution programs discussed in Chapter III, such a capability is not available. In fact, the HNORM model provides the response times directly for an interactive system with multiple job types.

Model Robustness

With the use of EQN theory, we have restricted ourselves to the assumption of exponential service distributions with FCFS service disciplines or to other service distributions with a rational Laplace transform and service disciplines of processor sharing (PS) or last-come-first-served preemptive-resume (LCFSPR) as discussed by Chandy et al [1972]. This restriction has made it possible to obtain solution forms for the EQN. However, these assumptions are not in the least way restrictive. The EQN models developed are extremely robust, particularly when we consider that we are not searching for details of network performance but for insights and trends. Other distributions may in fact be utilized. To give a specific example, different service distributions (other than exponential) were selected and used in the GPSS simulation. Recall that selection of service distributions is one of the advantages of the GPSS. The PIP and COMP were assigned hypo-exponential distributions of degree two [Hellerman and Conroy 75], expressed as a function of time, which is:

$$S(t) = [1 + 2\,(1 / u(i))\,t] ** [-2\,(1 / u(i))\,t] \quad \text{(VIII-1)}$$

where $S(t)$ = Pr[no job completes by time t]

and $1 - S(t)$ = Pr[a job completes by time t]

for i = P, C. Note, however, that the same mean service rates (u(i)) are to take on the same values used for the EQN models.

Such a distribution is one that is expected for the FIP and COMP functions in an interactive environment. The DBM and PEP were assigned hyper-exponential distributions for their expected processing function characteristics. Expressed as a function of time as was equation VIII-1:

$$S(t) = c^{**}[-2c(1/u(i))t] + (1-c)^{**}[-2(1-c)(1/u(i))t] \quad \text{(VIII-2)}$$

where c = .45 for i = D, P. Again note that the u(i) are the mean service rate values used in the EQN models. Table VIII-1 compares the use of balanced u(i) with exponential distributions to the same values of balanced u(i) with hypo- and hyper-exponential distributions. The GPSS simulations produce nearly identical results. It should also be noted that the service disciplines have all been FCFS. Such a combination of service disciplines and distributions is a clear violation of the EQN assumptions. Therefore, this example supports the conclusion that the solutions obtained using the EQN models will still yield excellent comparative results. Such comparisons and analysis are further enumerated in Buzen [1975] and Giammo [1976]. In our specific example, the performance criterion is response time, which is known to provide even less favorable comparisons than the performance criteria of utilization and thruput.

Results of Hyper-, Hypo-Exponential Distributions

GPSS Simulation

N	Distribution	Proportion of Types p(k)	Tr[N(k)] (sec)	Tr(N) (sec)
10	Exp	[.729,.150,.053,.067]	[.891,10.7,21.3,13.7]	4.32
	Hyper, Hypo	[.731,.147,.052,.068]	[.754,11.1,21.7,12.9]	4.21
			% error =	2.6
20	Exp	[.767,.129,.044,.059]	[1.53,17.7,32.9,20.3]	6.13
	Hyper, Hypo	[.763,.130,.044,.060]	[1.23,16.6,31.3,18.9]	5.65
			% error =	8.5
30	Exp	[.793,.113,.038,.054]	[3.16,27.2,47.8,29.3]	9.02
	Hyper, Hypo	[.790,.117,.038,.053]	[2.69,24.8,46.0,28.3]	8.32
			% error =	8.4

Note: The mean values of the distribution were those obtained for a balanced system (Chapter IV).

Table VIII-1

Model Comparisons

This section contains the results of each of the models developed, Basic Model (BM), Iterative Model (IM) and the Exact HNORM Model (HM), as well as the GPSS simulation verifications (S). Results are presented for two different sets of service rates. The balanced rates and those rates referred to as FEDSIM (used in the validation of the G Normalization constant at Federal Simulations as noted in Chapter VI). Both sets of rates are given for comparative purposes. Also, the GPSS run using the mean service rates for the balanced system but with hypo- and hyper-exponential distributions is presented. The model designations are shown as a function of B, F and H depending on the use of balanced or Fedsim rates with exponential distributions or balanced rates with the hyper-, hypo-exponential distributions. In all cases the zk = 15 sec and the job load distribution was taken such that N1 / N = 0.6, N2 / N = 0.2, N3 / N = 0.1 and N4 / N = 0.1 as was established in the parameter definition of the Basic Model and used for the Iterative and HNORM models.

Comparative Results

N=10, N1=6, N2=2, N3=N4=1

Model	Proportion of Types p(k)	Tr[N(k)] (sec)	Tr(N) (sec)	% err
BM(B)	[-- , -- , -- , --]	[-- , -- , -- , --]	4.50	4.2
IM(B)	[.873,.058,.027,.043]	[1.00,16.7,25.6,15.9]	3.22	25
HM(B)	[.731,.148,.053,.068]	[.907,11.1,21.6,13.6]	4.38	1.4
S(B)	[.729,.150,.053,.067]	[.891,10.7,21.3,13.7]	4.32	--
S(H)	[.731,.147,.052,.068]	[.754,11.1,21.7,12.9]	4.21	2.5
BM(F)	[-- , -- , -- , --]	[-- , -- , -- , --]	4.23	2.7
IM(F)	[.861,.071,.028,.040]	[1.03,13.4,24.4,17.2]	3.21	22
HM(F)	[.722,.159,.054,.065]	[.915,9.01,20.6,14.5]	4.14	0.5
S(F)	[.721,.159,.054,.064]	[.913,8.92,20.2,14.5]	4.12	--

Table VIII-2

Comparative Results

N=20, N1=12, N2=4, N3=N4=2

Model	Proportion of Types p(k)	Tr[N(k)] (sec)	Tr(N) (sec)	% err
BM(B)	[-- , -- , -- , --]	[-- , -- , -- , --]	7.64	25
IM(B)	[.840,.088,.832,.039]	[1.68,21.1,39.7,28.1]	5.58	9.0
HM(B)	[.766,.130,.044,.060]	[1.60,17.6,33.2,20.6]	6.20	1.1
S(B)	[.767,.129,.044,.059]	[1,53,17.7,32.9,20.3]	6.13	--
S(H)	[.763,.130,.044,.060]	[1.23,16.6,31.3,18.9]	5.65	7.8
BM(F)	[-- , -- , -- , --]	[-- , -- , -- , --]	7.26	22
IM(F)	[.819,.116,.026,.039]	[1.82,15.1,44.4,28.1]	5.49	7.6
HM(F)	[.753,.146,.045,.055]	[1.66,13.6,31.5,22.8]	5.92	0.3
S(F)	[.752,.146,.045,.055]	[1.69,13.6,31.2,22.8]	5.94	--

Table VIII-3

Comparative Results

N=30, N1=18, N2=6, N3=N4=3

Model	Proportion of Types p(k)	Tr[N(k)] (sec)	Tr(N) (sec)	% err
BM(B)	[-- , -- , -- , --]	[-- , -- , -- , --]	12.7	41
IM(B)	[.807,.110,.031,.052]	[3.51,27.8,59.4,32.7]	9.43	4.5
HM(B)	[.793,.115,.038,.053]	[3.22,26.8,47.9,30.1]	9.08	0.7
S(B)	[.793,.113,.038,.054]	[3.16,27.2,47.8,29.3]	9.02	--
S(H)	[.790,.117,.038,.053]	[2.69,24.8,46.0,28.3]	8.32	7.8
BM(F)	[-- , -- , -- , --]	[-- , -- , -- , --]	12.6	42
IM(F)	[.794,.126,.040,.055]	[3.80,23.8,53.4,37.5]	7.68	13
HM(F)	[.778,.134,.040,.049]	[3.48,20.8,45.6,34.2]	8.96	1.1
S(F)	[.778,.133,.038,.048]	[3.38,20.8,46.4,33.8]	8.86	--

Table VIII-4

As the models were developed, a comparison to GPSS for verification was made and a small discussion concerning the model's effectiveness was presented. Here we will make a comparison of the models and their effectiveness amongst themselves. As already noted, there is a definite region of acceptable effectiveness. Figure VIII-1 is a diagram indicating this fact.

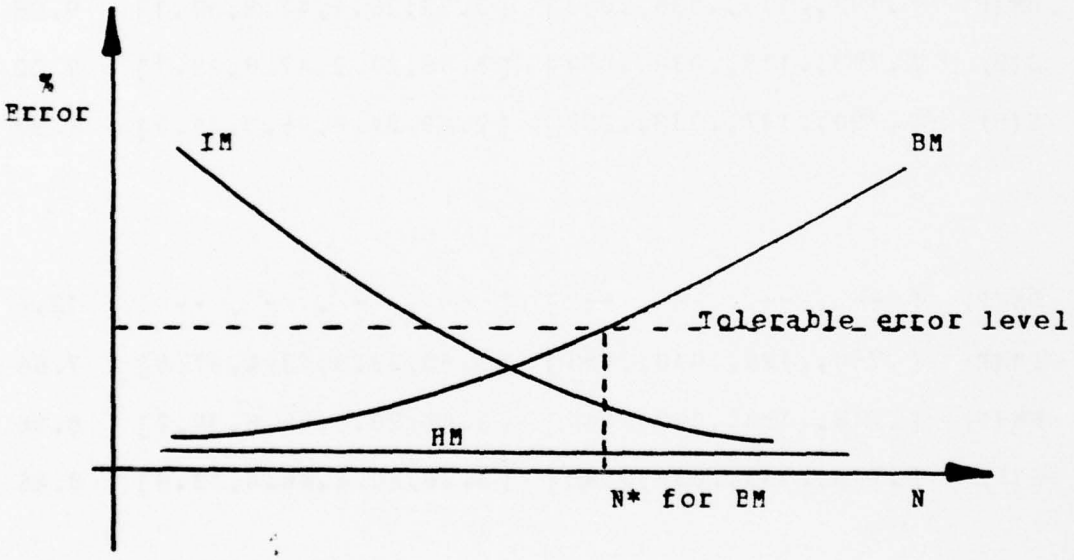


Figure VIII-1

This figure illustrates that the Basic Model (BM) will produce effective results when loaded below the saturation value, N*. The Iterative Model (IM) yields the best results when there is a large job load. Depending on the network, this region may well include job loads below the designed saturation. For instance, a network with a high N* may be modelled by the IM. The Exact H Model (HM) performs extremely well across the entire spectrum.

Recommended Use of the Models

Based on our previous discussion and Figure VIII-1 the following points are noted:

1. The Basic Model Formulation provides the basis for a balanced design criterion. Even though its response time results could exceed a designer's desired error tolerance level when more than N* users are active, the utilization results are sufficiently accurate to provide good design guidance.
2. When a quick, rough analysis is desired, the Basic Model is the best approach. This assumes that a General Purpose Program such as ASQ is available or one has been built from the basic program modules (Appendices A-F).
3. The Iterative Model (IM) can be used to analyze interactions between job types and their loading. By itself, it is not a particularly useful model. However, the G Computation Program established for its use is quite beneficial for the development of other program solution models.
4. The Exact HNORM Model (HM) provides more information and is more accurate in all regions than the BM but requires more computer execution time and storage requirements than the BM. The HM is also more versatile since one can consider individual job route

think times and it yields job route mean response times rather than just the composite response time.

Chapter IX: SUMMARY AND SUGGESTIONS FOR FUTURE WORK

Based on the techniques of Network Queuing Theory, three Exponential Queuing Network (EQN) models for the analysis and design of projected computer networks have been developed. The projected computer network must be parameterized in terms of functions in order to be modelled. Such functionalization is a natural result of the computer network structure and organization as noted in the examination of major networks. Functionalization such as computation, data base management, communications and terminal/access interface have already been considered in major military command and control systems. The EQN models of computer networks permit trade-off analysis of various network structures. Workload description is accommodated by permitting route definitions or circuit paths through the network which are composed of the defined processing functions. As a consequence, job typing or classification is implemented in the models. Additionally, a basic design technique is provided for the synthesis/determination of processing "computing power" to ensure optimal computer network load balance. Based on the major performance criterion of response time, a thorough evaluation or trade-off analysis of various computer network organizations and effectiveness may be conducted. The networks are considered to include all possible functions in an interactive mode. These models provide trends

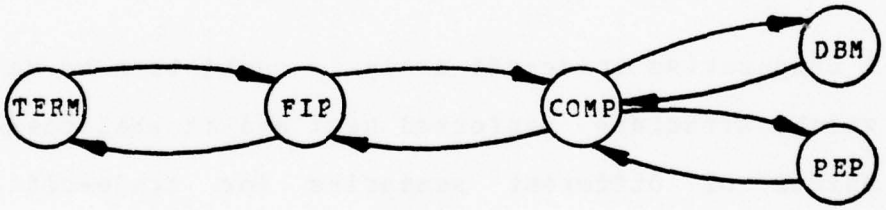
and insights into the performance areas. They are not intended to yield detailed information concerning performance, nor is it expected that the information available concerning the computer networks to be modelled will be available in detail.

A method for the computer network definition and description is presented in Chapter III. Each of the three EQN models developed, the Basic Model (BM) of Chapter IV, the Iterative Model (IM) of Chapter VI, and the Exact HNORM Model (HM) of Chapter VII employ the same methods of network definition and description. In Chapter V a classical event driven simulation programmed in GPSS has been developed in order to verify the models. This GPSS verification is necessary since we have developed models to represent projected computer networks which may be structured as a network of functionalized processors. Based on the verification using GPSS, the advantages, disadvantages, and the recommended uses for the models have been presented. The BM can provide quick, overall insights yielding its best results when the network is operating below saturation. The IM provides more detailed information and insights concerning job type or class interaction, but this is done at the expense of increased computation. The HM provides information on all job types and is accurate throughout the range of network loading.

In the development of the BM and IM, a general purpose single job type program analyzer, Duke ASQ, has been used.

Additionally, computer program modules for the solutions of the models are provided. These program modules enable the designer to tailor the model solution to his application or they can form the basis of a general purpose multi-job type program analyzer.

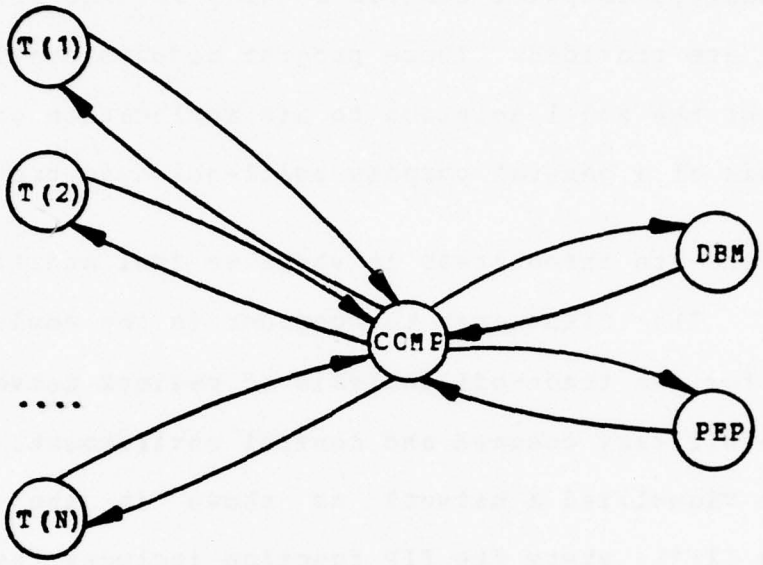
There are three areas in which we feel additional work is needed. The first area to consider is the employment of these models for the trade-off analysis of various network structures in the military command and control environment. For instance, we have visualized a network as shown in the diagram below (Figure IX-1), where the FIP function includes the capability of security or access control and TERM represents rather unintelligent terminals.



Computer Processing Network

Figure IX-1

On the other hand, we could consider a network structure such as the one in Figure IX-2 where the security control capability has been dispersed throughout the other elements of the network and the remaining capability of the FIP has been delegated to intelligent terminals.



Computer Processing Network

Figure IX-2

A comparative trade-off analysis could then be made to determine which structure performed best and at what cost. Although the number of different scenarios for trade-off analysis is virtually endless, we are able to selectively provide a quantitative basis upon which the decision-maker may resolve questions concerning network performance and, in the final analysis, its costs. This capability is particularly germaine in the area of military command and control where one needs to know in advance the expected response time of a network under a given load and the cost of achieving this response time.

A second area for future work is in the refinement of the program modules of Appendices A - F. These modules can be

further developed for computations of specific network structures or built into a general purpose system. The present implementation accommodates unlimited job typing or classes, numbers of nodes and numbers of jobs. The restrictions incurred are those of computational complexity and availability of computer resources.

A third area is the analysis of the GCT (or alternatively of the HCT) in terms of computational and storage efficiency and in terms of parallel processing algorithms. As noted earlier, when considering some of the verification runs with other ASQ systems, specifically the Duke ASQ and that at FEDSIM, it appears that our method of computation is more efficient. If so, further refinement of the program modules and their implementation could be extremely profitable.

APPENDIX A: PROGRAM FOR SOLUTION OF M EQUATIONS IN N UNKNOWNS

```
VISSOL:    PROC OPTIONS (MAIN);
   DCL MNODES FIXED BIN;
   DCL (P(MNODES,MNODES),PT(MNODES,MNODES)) FLOAT BIN CTL;
   DCL PAUG(MNODES,MNODES) FLOAT BIN CTL;
   DCL TEMPCOL(MNODES) FLOAT BIN CTL;
   DCL VSOL(MNODES-1) FLOAT BIN CTL;
   DCL (V(MNODES),MU(MNODES),X(MNODES)) FLOAT BIN CTL;

   /* P IS THE TRANSISTION PROBABILITY MATRIX.
      PT IS THE TRANSPOSE OF P. P IS SET UP WITH
      THE BALANCE EQUATIONS SUCH THAT THE V(I)
      FORM A LEFT EIGENVECTOR. THE V(I) WILL
      FORM A RIGHT EIGENVECTOR FOR PT. THE PAUG
      IS FORMED SUCH THAT THE RIGHT MOST COLUMN IS
      THE V VECTOR AND IS NORMALIZED SO THAT V(I)
      IS SET TO ONE.

       MNODES CORRESPONDS TO !M!    */


   GET LIST(MNODES);
   ALLOCATE P,PT,PAUG,TEMPCOL,VSOL,V,MU,X;
   P=0.0;PT=0.0;PAUG=0.0;VSOL=0.0;V=0.0;MU=0.0;X=0.0;
   GET LIST(P);

   PUT SKIP(3) DATA(MNODES);
   PUT SKIP(3) DATA(P);

   DO I=1 TO MNODES;
        PT(I,*)=P(*,I); END;

   PAUG=PT;
   DO I=1 TO MNODES;
        PAUG(I,I)=PT(I,I)-1.0; END;

   TEMPCOL(*)=PAUG(*,1);

   DO JC=2 TO MNODES;
        PAUG(*,JC-1)=PAUG(*,JC); END;

   PAUG(*,MNODES)=TEMPCOL(*);
   PUT SKIP(3) DATA(PAUG);
   FREE PT, TEMPCOL;

   CALL MATSOL(MNODES,MNODES,PAUG,VSOL);
   V(1)=1.0;
   DO I=2 TO MNODES;
```

```
     V(I)=VSOL(I-1); END;

/* DETERMINE THE X VALUES FOR NORM COMPUTATION */
/* NORMALIZE ON X(1)  */

     GET LIST (MU);

X(1)=V(1)/MU(1);
DO I=2 TO MNODES;
     X(I)=V(I)/MU(I)/X(1);
     END;
X(1)=1.0;
PUT SKIP(3) DATA (V);
PUT SKIP  DATA(MU);
PUT SKIP DATA(X);

/*    PROCEDURE FOR THE SOLUTION OF M EQUATIONS,
      N UNKNOWNS */
/*****************************************************/
MATSOL: PROC (M,N,XMAT,MSOL);
DECLARE (M,N) FIXED BINARY;
DCL XMAT(*,*) FLOAT BIN;
DCL MSOL(*) FLOAT BIN;

DYNAMIX:
BEGIN;


DECLARE (SIDE (M), TOP (N-1)) CHAR (1);
DCL BALPHA CHAR(12) INIT('ABCDEFGHIJKL');
DCL EALPHA CHAR(12) INIT('ZYXWVUTSRQPO');

DO I=1 TO N-1;
     TOP(I)=SUBSTR(EALPHA,N-I,1); END;
DO I=1 TO M;
     SIDE(I)=SUBSTR(BALPHA,I,1); END;
PUT PAGE;
PUT SKIP(3) LIST ('THE INITIAL MATRIX IS:');
CALL MPRINT (XMAT);
CALL SOLVE (XMAT,MSOL);


EXCHANGE: PROCEDURE (MATRIX,PIVROW,PIVCOL);
     DECLARE MATRIX(*,*) FLOAT BIN,
          (PIVROW,PIVCOL) FIXED BIN,
          (PIVOT,TEMP) FLOAT BIN,
          (ROWS,COLS,I,J) FIXED BIN;
     DECLARE TEMPA CHAR (1);

     /* DETERMINE NUMBER OF ROWS AND COLUMNS */
     ROWS = HBOUND (MATRIX,1);
     COLS = HBOUND (MATRIX,2);
```

```
      /* REFER TO THE PIVOT ELEMENT AS 'PIVOT' */
      PIVOT = MATRIX (PIVROW,PIVCOL);
      /* APPLY THE 'RECTANGLE RULE' */
      DO I = 1 TO PIVROW-1, PIVROW+1 TO ROWS;
           TEMP = MATRIX (I,PIVCOL);
           DO J = 1 TO PIVCOL-1, PIVCOL+1 TO COLS;
              MATRIX (I,J) = MATRIX (I,J) -
              TEMP * MATRIX (PIVROW,J) / PIVOT;
           END;
      END;
      /* CHANGE THE OLD PIVOT ROW */
      DO J = 1 TO PIVCOL-1, PIVCOL+1 TO COLS;
         MATRIX (PIVROW,J) = -MATRIX (PIVROW,J) / PIVOT;
      END;
      /* CHANGE THE OLD PIVOT COLUMN */
      DO I = 1 TO PIVROW-1, PIVROW+1 TO ROWS;
         MATRIX (I,PIVCOL) = MATRIX (I,PIVCOL) / PIVOT;
      END;
      /* CHANGE THE PIVOT */
      MATRIX (PIVROW,PIVCOL) = 1 / PIVOT;
      /* ALTER THE 'TOP' AND 'SIDE' ARRAYS APPROPRIATELY */
      TEMPA = SIDE (PIVROW);
      SIDE (PIVROW) = TOP (PIVCOL);
      TOP (PIVCOL) = TEMPA;
END EXCHANGE;


MPRINT: PROCEDURE (MATRIX);
      DCL MATRIX (*,*) FLOAT BIN,
           (ROWS,COLS,I,J) FIXED BIN;

      /* DETERMINE NUMBER OF ROWS AND COLUMNS */
      ROWS = HBOUND (MATRIX,1);
      COLS = HBOUND (MATRIX,2);
      /* PRINT THE 'TOP' ARRAY */
      PUT SKIP EDIT (' ') (X(4),A);
      DO I=1 TO COLS -1;
           PUT EDIT (TOP (I),' ') (X(8),A,X(6),A);
           END;
      PUT SKIP;
      /* PRINT THE MATRIX LINE BY LINE */
      DO I=1 TO ROWS;
           /* EACH ROW OF THE MATRIX STARTS
              ON A NEW OUTPUT LINE */
           PUT SKIP EDIT (SIDE(I)) (X(4),A);
           DO J=1 TO COLS;
           /* OMIT THE WORD 'SKIP' - SO WE WILL
           PRINT THE ENTIRE ROW ON A NEW OUTPUT LINE */
                PUT EDIT (MATRIX (I,J))(F(16,6));
           END;
      END;
      PUT SKIP;
```

```
END MPRINT;


SOLVE: PROC(MAT,SOL);
     DCL MAT(*,*) FLOAT BIN,
          SOL(*) FLOAT BIN,
          (SUM,ANS) FLOAT BIN,
          (R,C,MIN,K,KK,P,Q) FIXED BIN;

     PUT SKIP LIST ('ENTER THE SOLVE ROUTINE');
     R=HBOUND(MAT,1);
     C=HBOUND(MAT,2);
     MIN = C - 1;
     IF R < MIN THEN MIN=R;

   BEGIN;
     DCL (TEMP(R,C)) FLOAT BIN;
     DCL TEMPTOP (N-1) CHAR(1);
     TEMP=MAT;
     TEMPTOP=TOP;
     DO K=1 TO MIN;
          CALL CHOOSE (MAT,P,Q);
          IF P=0 THEN DO;
            PUT SKIP LIST
            ('SOLVE COULD NOT DO ALL INTENDED EXCHANGES');
            GO TO NOMORE;
            END;
          CALL EXCHANGE (MAT,P,Q);
     END;

     NOMORE:
     PUT SKIP EDIT('THE SOLUTION FOUND IS: ')(A);
     SOL=0;
     DO K=1 TO N-1;
          DO KK=1 TO M;
            IF TEMPTOP(K) = SIDE(KK) THEN DO;
            SOL(K) = MAT(KK,C);
            KK=M;
            END;
          END;
          PUT EDIT (TEMPTOP(K),' = ',SOL(K))
          (X(5),A,A,F(15,6))
     END;


     PUT SKIP(2) LIST('VERIFY OUR RESULTS');
     PUT SKIP EDIT('THERE ARE ',R,' EQUATIONS TO SATISFY')
          (A,F(4),A);
     PUT SKIP;
```

```
          DO K=1 TO R;
               SUM = 0;
               DO KK=1 TO N-1;
                 SUM=SUM+TEMP(K,KK)*SOL(KK);
               END;
               ANS = -TEMP(K,C);
               PUT SKIP EDIT(SUM,' SHOULD = ',ANS)
                 (F(16,6),A,F(16,6));
          END;

        END; /* OF BEGIN BLOCK IN SOLVE*/

      END SOLVE;


      CHOOSE: PROC(MAT,PR,PC);
          DCL MAT(*,*) FLOAT BIN,
               (PR,PC) FIXED BIN,
               MAX FLOAT BIN,
               (I,J) FIXED BIN;
          MAX = 1E-6;
          PR = 0;
          DO I=1 TO M;
               IF SIDE(I) < 'M' THEN
                 /*THIS IS AN ELGIBLE ROW */
                 IF ABS(MAT(I,J)) > MAX THEN
                    /* THIS IS THE BEST PIVOT SO FAR*/
                    DO;
                    MAX=ABS(MAT(I,J));
                    PR=I;
                    PC=J;
                    END;
          END;
      END CHOOSE;


      PUT SKIP DATA(MSOL);


      END DYNAMIX:

      /*       END OF PROC M EA, N UKN    */
      /*************************************************/

  END VISSOL;
```

APPENDIX B: EVENT DRIVEN SIMULATION USING GPSS

```
        SIMULATE
*
*       INITIALIZATION OF MODEL PARAMETERS
*
*       TIMER CONTROLS AND RN SEEDS
        RMULT       600,600,600
        INITIAL     X$TIMER,21600000/X$TINCR,7200000
*
*       PROCESSOR MEAN SERVICE TIMES
        INITIAL     XH$MFIP,600
        INITIAL     XH$MCOMP,1375
        INITIAL     XH$MDBM,4125
        INITIAL     XH$MPEP,6600
        INITIAL     XH$MTT1,15000
        INITIAL     XH$MTT2,15000/XH$MTT3,15000/XH$MTT4,15000
        INITIAL     XH$NRT1,6/XH$NRT2,2/XH$NRT3,1/XH$NRT4,1
*
        INITIAL     XH$NUSER,10
*
 RTCNT  MATRIX      MX,2,4          MATRIX FOR ROUTE COUNTS
*                                    ROW 1 - COUNT
*                                    ROW 2 - NORMALIZED AT 1000
 ENT    MATRIX      MX,5,5
*
*       FUNCTION DEFINITIONS
*
 EXPD   FUNCTION    RN1,C24      FIP SERVICE TIME DIST
0,0/.1,.104/.2,.222/.3,.355/.4,.509/.5,.69/.6,.915/.7,1.2
*
*
*
*       BRANCH CONDITIONS ON BOOLEAN VARIABLES
*
 BTT    BVARIABLE   P3'GE'1+P2'E'1
 BDBM   BVARIABLE   (P2'E'2+P2'E'3)*(P3'E'0)
 BPEP   BVARIABLE   (P2'E'3*P3'E'1)+(P2'E'4*P3'E'0)
 BFIP   BVARIABLE   (P2'NE'3*P3'E'1)+(P2'E'3*P3'E'2)
*
*
*       VARIABLES
 VINIT  FVARIABLE   X$TIMER/3600000
 VRUN   FVARIABLE   X$TINCR/3600000
 SUM1   VARIABLE    MX$RTCNT(1,1)+MX$RTCNT(1,2)
 SUM2   VARIABLE    MX$RTCNT(1,3)+MX$RTCNT(1,4)
 SUM    VARIABLE    V$SUM1+V$SUM2
 PER1   FVARIABLE   (MX$RTCNT(1,1)/V$SUM)*1000
 PER2   FVARIABLE   (MX$RTCNT(1,2)/V$SUM)*1000
 PER3   FVARIABLE   (MX$RTCNT(1,3)/V$SUM)*1000
```

```
 TCNT   FVARIABLE  MX$ENT(1,1)+MX$ENT(1,2)+MX$ENT(1,3)
                    +MX$ENT(1,4)
 FCNT   FVARIABLE  MX$ENT(2,1)+MX$ENT(2,2)+MX$ENT(2,3)
                    +MX$ENT(2,4)
 CCNT   FVARIABLE  MX$ENT(3,1)+MX$ENT(3,2)+MX$ENT(3,3)
                    +MX$ENT(3,4)
 DCNT   FVARIABLE  MX$ENT(4,1)+MX$ENT(4,2)+MX$ENT(4,3)
                    +MX$ENT(4,4)
 PCNT   FVARIABLE  MX$ENT(5,1)+MX$ENT(5,2)+MX$ENT(5,3)
                    +MX$ENT(5,4)
 PER4   FVARIABLE  (MX$RTCNT(1,4)/V$SUM)*1000
 MRESA  FVARIABLE  TB$RESR1*TC$RESR1+TB$RESR2*TC$RESR2
 MRESB  FVARIABLE  TB$RESR3*TC$RESR3+TB$RESR4*TC$RESR4
 MDEN   FVARIABLE  TC$RESR1+TC$RESR2+TC$RESR3+TC$RESR4
 MREST  FVARIABLE  (V$MRESA+V$MRESB)/V$MDEN
*
*       STORAGE, IE, MUITIPLE SERVERS
*
        STORAGE    S$TERMS,10
*
*       TABLE DEFINITIONS
 RESR1  TABLE       MP1,100,100,100 RT ROUTE 1
 RESR2  TABLE       MP1,2000,500,100 RT ROUTE 2
 RESR3  TABLE       MP1,2000,500,100 RT ROUTE 3
 RESR4  TABLE       MP1,2000,500,100 RT ROUTE 4
*
*****************************************************************
*
*       MODEL SEGMENT FOR ROUTE 1
*
*       P1:  RESPONSE TIME
*       P2:  ROUTE NUMBER (1,2,3,4)
*       P3:  IN - 0
*            OUT -   1 FOR ROUTES 2 AND 4
*                    3 FOR ROUTE 3
*       P4:  MEAN THINK TIME
*
*
 ENR1   GENERATE    ,,,XH$NRT1,,6,F
        ASSIGN      2,1
        TRANSFER    ,TERM
*
*
 ENR2   GENERATE    ,,,XH$NRT2,,6,F
        ASSIGN      2,2
        TRANSFER    ,TERM
```

```
*
*
 ENR3  GENERATE    ,,,XH$NRT3,,6,F
       ASSIGN      2,3
       TRANSFER    ,TERM
*
*
 ENR4  GENERATE    ,,,XH$NRT4,,6,F
       ASSIGN      2,4
*
*
*
*
*****************************************************************
*
*       MODEL SEGMENT FOR THE COMPOSITE NETWORK
*
 TERM  TEST E      P2,1,OTHER
       ASSIGN      4,XH$MTT1
       TRANSFER    ,ENTER
 OTHER ASSIGN      4,XH$MTT2
 ENTER QUEUE       TERMQ          TERM STAGE
       MSAVEVALUE  RTCNT+,1,P2,1,MX
       MSAVEVALUE  ENT+,1,P2,1,MX
       ENTER       TERMS
       ADVANCE     P4
       LEAVE       TERMS
       DEPART      TERMQ
       ASSIGN      3,0            DESIGNATE AS INWARD TO THE NETWORK
       MARK        1              ENT INTO THE NETWORK
*
*
*       FIP PROCESSOR
 FIP   QUEUE       FIPQ           FIP STAGE
       MSAVEVALUE  ENT+,2,P2,1,MX
       SEIZE       FIPF
       ADVANCE     XH$MFIP,FN$EXPD
       RELEASE     FIPF
       DEPART      FIPQ
       TEST E      BV$BTT,1,COMP
       TEST E      P2,1,BPT1
*
*       ROUTE RESPONSE TIME TABULATIONS
       TABULATE    RESR1
       TRANSFER    ,TERM
```

```
*
 BPT1   TEST E      P2,2,BPT2
        TABULATE    RESR2
        TRANSFER    ,TERM
*
 BPT2   TEST E      P2,3,BPT3
        TABULATE    RESR3
        TRANSFER    ,TERM
*
 BPT3   TEST E      P2,4,BPT4
        TABULATE    RESR4
        TRANSFER    ,TERM
*
 BPT4   TRANSFER    ,TERM
*
*
*
*       COMP PROCESSOR
 COMP   QUEUE       COMPQ
        MSAVEVALUE  ENT+,3,P2,1,MX
        SEIZE       COMPF
        ADVANCE     XH$MCCMP,FN$EXPD
        RELEASE     COMPF
        DEPART      COMPQ
        TEST E      BV$BDBM,0,DBM
        TEST E      BV$BPEP,0,PEP
        TEST E      BV$BFIP,0,FIP
*
*
*       DBM PROCESSOR
 DBM    QUEUE       DBMQ
        MSAVEVALUE  ENT+,4,P2,1,MX
        SEIZE       DBMF
        ADVANCE     XH$MDBM,FN$EXPD
        RELEASE     DBMF
        DEPART      DBMQ
        ASSIGN      3+,1        INCREMENT DIRECTION COUNT
        TRANSFER    ,COMP
*
*
*       PEP PROCESSOR
 PEP    QUEUE       PEPQ
        MSAVEVALUE  ENT+,5,P2,1,MX
        SEIZE       PEPF
        ADVANCE     XH$MPEP,FN$EXPD
        RELEASE     PEPF
        DEPART      PEPQ
        ASSIGN      3+,1        INCREMENT DIRECTION COUNT
        TRANSFER    ,COMP
```

```
*
*
*       MODEL  SEGMENT FOR TIMER RUN CONTROL
*
 CNTRL  GENERATE    X$TIMER
        MSAVEVALUE  RTCNT,2,1,V$PER1,MX NORMALIZED VALUES IN ROW 2
        MSAVEVALUE  RTCNT,2,2,V$PER2,MX
        MSAVEVALUE  RTCNT,2,3,V$PER3,MX
        MSAVEVALUE  RTCNT,2,4,V$PER4,MX
        SAVEVALUE   CREST,V$MREST,XL
        MSAVEVALUE  ENT,1,5,V$TCNT,MX
        MSAVEVALUE  ENT,2,5,V$PCNT,MX
        MSAVEVALUE  ENT,3,5,V$CCNT,MX
        MSAVEVALUE  ENT,4,5,V$DCNT,MX
        MSAVEVALUE  ENT,5,5,V$PCNT,MX
        SAVEVALUE   TINIT,V$VINIT,XL
        SAVEVALUE   TRUN,V$VRUN,XL
        TERMINATE   1
*
*       CONTROL CARDS
        NOXREF
*
        LOAD        DAG06,DAG07
        START       1
        INITIAL     MX1(1-2,1-4),0
        INITIAL     MX2(1-5,1-5),0
        RESET
 CNTRL  GENERATE    X$TINCR
        START       1
*                                RECONFIGURE FOR 20 JOBS
        INITIAL     XH$NUSER,20
        INITIAL     XH$NRT1,12/XH$NRT2,4/XH$NRT3,2/XH$NRT4,2
        INITIAL     X$TIMER,43200000/X$TINCR,14400000
        CLEAR       XH1-XH13,X1-X2
        STORAGE     S$TERMS,20
 CNTRL  GENERATE    X$TIMER
        START       1
        INITIAL     MX1(1-2,1-4),0
        INITIAL     MX2(1-5,1-5),0
        RESET
 CNTRL  GENERATE    X$TINCR
        START       1
*                                RECONFIGURE FOR 30 JOBS
        INITIAL     XH$NUSER,30
        INITIAL     XH$NRT1,18/XH$NRT2,6/XH$NRT3,3/XH$NRT4,3
        INITIAL     X$TIMER,64800000/X$TINCR,21600000
        CLEAR       XH1-XH13,X1-X2
        STORAGE     S$TERMS,30
 CNTRL  GENERATE    X$TIMER
        START       1
        INITIAL     MX1(1-2,1-4),0
        INITIAL     MX2(1-5,1-5),0
```

```
        RESET
CNTRL   GENERATE    X$TINCR
        START       1
        REPORT
        EJECT
        TEXT        NUMBER OF JOBS IS #XH$NUSER,2/XXX#
        TEXT        INITIALIZATION HOURS ARE #XL$TINIT,2/XXX.XX#
        TEXT        DATA COLLECTION HOURS ARE #XL$TRUN,2/XXX.XX#
        TEXT        THINK TIMES ARE - A: #XH$MTT1,2/XXXXXX#
18      TEXT         B: #XH$MTT2,2/XXXXXX#
18      TEXT         C: #XH$MTT3,2/XXXXXX#
18      TEXT         D: #XH$MTT4,2/XXXXXX#
        TEXT        SERVICE TIMES(MSEC) ARE -
26      TEXT         FIP:  EXP #XH$MFIP,2/XXXXXX#
26      TEXT         COMP: EXP #XH$MCOMP,2/XXXXXX#
26      TEXT         DBM:  EXP #XH$MDBM,2/XXXXXX#
26      TEXT         PEP:  EXP #XH$MPEP,2/XXXXXX#
        SPACE       1
FMS     TITLE       RTCNT,THE JOB COUNT BY ROUTE
        SPACE       3
        TEXT        COMPOSITE MEAN R TIME IS#XL$CREST,2/XXXXX.XXX#
TAB     INCLUDE     T1-T4/1,2,3,4
        SPACE       3
QUE     TITLE       ,QUEUE STATISTICS
QUE     INCLUDE     Q1-Q5/1,3,7
        SPACE       3
FAC     TITLE       ,PROCESSOR STATISTICS
FAC     INCLUDE     F1-F4/1,2,3,4
        SPACE       3
FMS     TITLE       ENT,PROCESSOR ENT COUNT BY ROUTE
        EJECT
*
*       FOLLOWING DENSITY PLOTS ARE OPTIONAL
*
TAB     TITLE       RESR1,ROUTE 1 RESPONSE TIME FREQUENCY
        EJECT
        GRAPH       TF,RESR1
        ORIGIN      53,10
        X           ,1,0,100,1,100,NO
        Y           0,10,50,1
30      STATEMENT   1,31,ROUTE 1 RESPONSE TIME FREQUENCY
        ENDGRAPH
        EJECT
TAB     TITLE       RESR2,ROUTE 2 RESPONSE TIME FREQUENCY
        EJECT
        GRAPH       TF,RESR2
        ORIGIN      53,10
        X           ,1,0,2000,1,100,NO
        Y           0,1,50,1
30      STATEMENT   1,31,ROUTE 2 RESPONSE TIME FREQUENCY
        ENDGRAPH
        EJECT
```

```
TAB     TITLE       RESR3,ROUTE 3 RESPONSE TIME FREQUENCY
        EJECT
        GRAPH       TF,RESR3
        CRIGIN      53,10
        X           ,1,0,2000,1,100,NO
        Y           0,1,50,1
30      STATEMENT   1,31,ROUTE 3 RESPONSE TIME FREQUENCY
        ENDGRAPH
        EJECT
TAB     TITLE       RESR4,ROUTE 4 RESPONSE TIME FREQUENCY
        EJECT
        GRAPH       TF,RESR4
        CRIGIN      53,10
        X           ,1,0,2000,1,100,NO
        Y           0,1,50,1
30      STATEMENT   1,31,ROUTE 4 RESPONSE TIME FREQUENCY
        ENDGRAPH
        EJECT
        END
```

APPENDIX C: G COMPUTATION PROGRAM FOR COMPUTER NETWORKS
WITH MULTIPLE JOB TYPES

```
NORM:   PROC OPTIONS (MAIN);

   /* COMPUTES THE NORMALIZATION CONSTANTS FOR M NODES AND
   UP TO FOUR JOB CLASSES:
        A - 1ST JOB CLASS
        NA - NUMBER OF A JOBS
        B - 2ND JOB CLASS
        NB - NUMBER OF B JOBS
        C - 3RD JOB CLASS
        NC - NUMBER OF C JOBS
        D - 4TH JOB CLASS
        ND - NUMBER OF D JOBS

   G(NA,NB,NC,ND) IS AN ARRAY FOR THE LAST OR HIGHEST VALUE OF
        M AND IS MAINTAINED IN STORAGE. THE M IS IMPLIED.

   XA(M),XB(M),XC(M),XD(M) ARE THE RESPECTIVE, RELATIVE
        UTILIZATIONS FOR JOB CLASSES A,B,C,D.  */

   /*  INPUT IS FREEFORMAT:
        NO OF CLASSES, NCLASS;
        M, THE NUMBER OF NODES;
        NO OF INDIVIDUAL CLASSES, NA,NB,NC,ND;
        RELATIVE UTILIZATIONS, XA,XB,XC,XD.
             THESE ARE VECTORS OF LENGTH M;
        SERVICE RATES AT THE NODES (1,2,3,4);
        A FLAG FOR PRINTER OUTPUT OF THE G, -Y OR N;


   A OUTPUT FILE, G4OUT, MUST BE ALLOCATED FOR THE
             G MATRIX.    */

   DCL(LA,LB,LC,LD) FIXED BIN;
   DCL  (M,NCLASS,NA,NB,NC,ND) FIXED BIN ;
   DCL  (IA,IB,IC,ID) FIXED BIN;

   GET LIST(NCLASS,M);
   PUT SKIP DATA(NCLASS,M);

   /* DETERMINE THE NUMBER OF CLASSES */

   IF NCLASS=1 THEN GO TO FINISH;
   IF NCLASS=2 THEN GO TO FINISH;
   IF NCLASS=3 THEN GO TO FINISH;
   IF NCLASS=4 THEN GO TO CLASS4;
```

```
   /* ESTABLISHED FOR 4 CLASSES.
   OTHERS ARE SUBCASES OF 4 AND ARE
   NOT INCLUDEED IN THIS LISTING.  */


CLASS4:      /* CASE OF FOUR JOB CLASSES */

   GET LIST(NA,NB,NC,ND);
   PUT SKIP DATA(NA,NB,NC,ND);

   BEGIN;
   DCL (XA(M),XB(M),XC(M),XD(M)) FLOAT BIN;
   DCL(G(0:NA,0:NB,0:NC,0:ND), GC(0:NC,0:M), GD(0:ND,0:M),
    GA(0:NA,0:M), GB(0:NB,0:M)) FLOAT BIN;
   DCL (GAB(0:NA,0:NB,0:M),GAC(0:NA,0:NC,0:M),
        GBC(0:NB,0:NC,0:M)) FLOAT BIN;
   DCL (GAD(0:NA,0:ND,0:M),GBD(0:NB,0:ND,0:M),
        GCD(0:NC,0:ND,0:M)) FLOAT BIN;
   DCL (GABC(0:NA,0:NB,0:NC,0:M),
        GABD(0:NA,0:NB,0:ND,0:M),
        GACD(0:NA,0:NC,0:ND,0:M),
        GBCD(0:NB,0:NC,0:ND,0:M)) FLOAT BIN;
   DCL (U(M),UA(M),UB(M),UC(M),UD(M)) FLOAT BIN;
   DCL (T(M),TA(M),TB(M),TC(M),TD(M)) FLOAT BIN;
   DCL MU(M) FLOAT BIN;
   DCL (TSA,TSB,TSC,TSD) FLOAT BIN;
   DCL GPRNT CHAR(1);
GEDIT:  FORMAT(A,F(NA/10+1),A,F(NB/10+1),A,F(NC/10+1),
        A,F(ND/10+1),A,E(15,8),SKIP);
TEDIT:  FORMAT(A,SKIP,(3)(A,F(2),X(3)));
   GET LIST(XA,XB,XC,XD);
   PUT SKIP DATA(XA,XB,XC,XD);
   GET LIST(MU);
   PUT SKIP DATA(MU);
   GET LIST (GPRNT);
   IF NA=0 THEN PUT SKIP LIST('CLASS ERROR');

   /* INITIALIZE THE G */
   G=0.0; GA=0.0; GB=0.0; GC=0.0; GD=0.0;
   GAB=0.0; GAC=0.0;GBC=0.0;
   GAD=0.0; GBD=0.0; GCD=0.0;
   GABC=0.0; GABD=0.0;GACD=0.0;GBCD=0.0;
   CALL ONEJOB(NA,XA,GA);
   CALL ONEJOB(NB,XB,GB);
   CALL ONEJOB(NC,XC,GC);
   CALL ONEJOB(ND,XD,GD);
```

```
CALL TWOJOB(NA,NB,XA,XB,GAB,GB,GA);
CALL TWOJOB(NA,NC,XA,XC,GAC,GC,GA);
CALL TWOJOB(NB,NC,XB,XC,GBC,GC,GB);
CALL TWOJOB(NA,ND,XA,XD,GAD,GD,GA);
CALL TWOJOB(NB,ND,XB,XD,GBD,GD,GB);
CALL TWOJOB(NC,ND,XC,XD,GCD,GD,GC);


CALL THREJOB(NA,NB,NC,XA,XB,XC,GABC,GBC,GAC,GAB);
CALL THREJOB(NA,NB,ND,XA,XB,XD,GABD,GBD,GAD,GAB);
CALL THREJOB(NA,NC,ND,XA,XC,XD,GACD,GCD,GAD,GAC);
CALL THREJOB(NB,NC,ND,XB,XC,XD,GBCD,GCD,GBD,GBC);

/* COMPUTE THE G */

CALL FOURJOB(NA,NB,NC,ND,XA,XB,XC,XD,
     G,GBCD,GACD,GABD,GABC);

NT=NA+NB+NC+ND;
IF GPRNT='Y' THEN CALL PUTG(NA,NB,NC,ND,G);
     ELSE DO;
     PUT FILE(G4OUT) EDIT(NA,NB,NC,ND)((4)(F(2),X(2)),SKIP);
     PUT FILE(G4OUT) EDIT(G)((4)(E(15,8),X(3)),SKIP);
     END;
PUT SKIP(2) EDIT('G(',NA-1,',',NB,',',NC,',',ND,')= ',
     G(NA-1,NB,NC,ND))(R(GEDIT));
PUT SKIP(2) EDIT('G(',NA,',',NB-1,',',NC,',',ND,')= ',
     G(NA,NB-1,NC,ND))(R(GEDIT));
PUT SKIP(2) EDIT('G(',NA,',',NB,',',NC-1,',',ND,')= ',
     G(NA,NB,NC-1,ND))(R(GEDIT));
PUT SKIP(2) EDIT('G(',NA,',',NB,',',NC,',',ND-1,')= ',
     G(NA,NB,NC,ND-1))(R(GEDIT));
PUT SKIP(2) EDIT('G(',NA,',',NB,',',NC,',',ND,')= ',
     G(NA,NB,NC,ND))(R(GEDIT));
/* COMPUTE THE UTILIZATION AND THRUPUT FACTORS */
DO I=1 TO M;
     UA(I)=XA(I)*G(NA-1,NB,NC,ND)/G(NA,NB,NC,ND);
     TA(I)=UA(I)*MU(I);
     UB(I)=XB(I)*G(NA,NB-1,NC,ND)/G(NA,NB,NC,ND);
     TB(I)=UB(I)*MU(I);
     UC(I)=XC(I)*G(NA,NB,NC-1,ND)/G(NA,NB,NC,ND);
     TC(I)=UC(I)*MU(I);
     UD(I)=XD(I)*G(NA,NB,NC,ND-1)/G(NA,NB,NC,ND);
     TD(I)=UD(I)*MU(I);
     U(I)= UA(I)+UB(I)+UC(I)+UD(I);
     T(I)=TA(I)+TB(I)+TC(I)+TD(I);
     PUT SKIP(2) EDIT('NODE ',I,': ',' U= ',U(I),
          ' UA= ',UA(I),' UB= ',UB(I),' UC= ',UC(I),' UD= ',
          UD(I),' T= ',T(I),' TA= ',TA(I),' TB= ',TB(I),
          ' TC= ',TC(I),' TD= ',TD(I))
          (A,F(2),A,SKIP,(2)((5)(X(3),A,F(5,3)),SKIP));
END;
```

```
/* COMPUTE BRANCHING RATICS */

PA=TA(1)/T(1);
PB=TB(1)/T(1);
PC=TC(1)/T(1);
PD=TD(1)/T(1);
PUT SKIP(3) EDIT('THRUPUT BRANCHING IS:',
     ' PA= ',PA,' PB= ',PB,' PC= ',PC,' PD= ',PD)
     (A,SKIP,(4)(X(3),A,F(6,3)));

/* COMPUTE SHORT CIRCUIT THRUPUT */

TSA=G(NA-1,NB,NC,ND)/G(NA,NB,NC,ND);
TSB=G(NA,NB-1,NC,ND)/G(NA,NB,NC,ND);
TSC=G(NA,NB,NC-1,ND)/G(NA,NB,NC,ND);
TSD=G(NA,NB,NC,ND-1)/G(NA,NB,NC,ND);
PUT SKIP(3) EDIT('SHORT CIRCUIT THRUPUT FOLLCWS:',
     ' TSA= ',TSA,' TSB= ',TSB,' TSC= ',TSC,
     ' TSD= ',TSD)
     (A,SKIP,(4)(X(3),A,F(6,3)));


/*COMPUTE THE THRUPUT AS A FUNCTION CF THE
  NUMBER OF JOBS IN EACH JCB CLASS WITH THE
  OTHER JOBS HELD FIXED.*/

PUT SKIP(3) EDIT('THRUPUT VS. AJOBS',
     'NB= ',NB,'NC= ',NC,'ND= ',ND)(R(TEDIT));
DO IA=1 TO NA;
     TSA=G(IA-1,NB,NC,ND)/G(IA,NB,NC,ND);
     PUT SKIP EDIT('NA= ',IA,'TSA= ',TSA)
          (A,F(2),X(3),A,F(7,3));
END;

PUT SKIP(3) EDIT('THRUPUT VS. BJOBS',
     'NA= ',NA,'NC= ',NC,'ND= ',ND)(R(TEDIT));
DO IB=1 TO NB;
     TSB=G(NA,IB-1,NC,ND)/G(NA,IB,NC,ND);
     PUT SKIP EDIT('NB= ',IB,'TSB= ',TSB)
          (A,F(2),X(3),A,F(7,3));
END;

PUT SKIP(3) EDIT('THRUPUT VS. CJOBS',
     'NA= ',NA,'NB= ',NB,'ND= ',ND)(R(TEDIT));
DO IC=1 TO NC;
     TSC=G(NA,NB,IC-1,ND)/G(NA,NB,IC,ND);
     PUT SKIP EDIT('NC= ',IC,'TSC= ',TSC)
          (A,F(2),X(3),A,F(7,3));
END;

PUT SKIP(3) EDIT('THRUPUT VS. DJOBS',
     'NA= ',NA,'NB= ',NB,'NC= ',NC)(R(TEDIT));
```

```
    DO ID=1 TO ND;
         TSD=G(NA,NB,NC,ID-1)/G(NA,NB,NC,ID);
         PUT SKIP EDIT('ND= ',ID,'TSD= ',TSD)
              (A,F(2),X(3),A,F(7,3));
    END;

    END; /* BEGIN BLOCK CLASS3 */
    GO TO FINISH;

ONEJOB: PROC(N1,X1,G1);
    DCL N1 FIXED BIN;
    DCL X1(*) FLOAT BIN;
    DCL G1(*,*) FLOAT BIN;

    G1(0,*)=1.0;

    DO LM=1 TO M;
         DO LA=1 TO N1;
              G1(LA,LM)=G1(LA,LM-1) + X1(LM) * G1(LA-1,LM);
         END;
    END;
    END ONEJOB;


TWOJOB: PROC(N1,N2,X1,X2,G2,G2Z1,G2Z2);
    DCL (N1,N2) FIXED BIN;
    DCL (X1(*),X2(*)) FLOAT BIN;
    DCL (G2(*,*,*),G2Z1(*,*),G2Z2(*,*)) FLOAT BIN;
    DO LM=1 TO M;
         G2(*,0,LM)=G2Z2(*,LM);
         G2(0,*,LM)=G2Z1(*,LM);
         DO LB=1 TO N2;
              DO LA=1 TO N1;
                   G2(LA,LB,LM) = G2(LA,LB,LM-1) +
                      X1(LM)*G2(LA-1,LB,LM)
                    + X2(LM)*G2(LA,LB-1,LM);
              END;
         END;
    END;
    END TWOJOB;

THREJOB:     PROC(N1,N2,N3,X1,X2,X3,G3,G3Z1,G3Z2,G3Z3);
    DCL (N1,N2,N3) FIXED BIN;
    DCL (X1(*),X2(*),X3(*)) FLOAT BIN;
    DCL (G3(*,*,*,*),G3Z1(*,*,*),
         G3Z2(*,*,*),G3Z3(*,*,*)) FLOAT BIN;

    DO LM=1 TO M;
         G3(0,*,*,LM)=G3Z1(*,*,LM);
         G3(*,0,*,LM)=G3Z2(*,*,LM);
         G3(*,*,0,LM)=G3Z3(*,*,LM);
```

```
          DO LC=1 TO N3;
               DO LB=1 TO N2;
                    DO LA=1 TO N1;

                    G3(LA,LB,LC,LM)=G3(LA,LB,LC,LM-1)
                      +X1(LM)*G3(LA-1,LB,LC,LM)
                      +X2(LM)*G3(LA,LB-1,LC,LM)
                      +X3(LM)*G3(LA,LB,LC-1,LM);

                    END;
               END;
          END;
     END;
     END THREJOB;

FOURJOB:      PROC(N1,N2,N3,N4,X1,X2,X3,X4,
          G4,G4Z1,G4Z2,G4Z3,G4Z4);
     DCL(N1,N2,N3,N4) FIXED BIN;
     DCL(X1(*),X2(*),X3(*),X4(*)) FLOAT BIN;
     DCL(G4(*,*,*,*),G4Z1(*,*,*,*),G4Z2(*,*,*,*),
          G4Z3(*,*,*,*),G4Z4(*,*,*,*)) FLOAT BIN;

     DO LM=1 TO M;
          G4(0,*,*,*)=G4Z1(*,*,*,LM);
          G4(*,0,*,*)=G4Z2(*,*,*,LM);
          G4(*,*,0,*)=G4Z3(*,*,*,LM);
          G4(*,*,*,0)=G4Z4(*,*,*,LM);

          DO LD=1 TO N4;
               DO LC=1 TO N3;
                    DO LB=1 TO N2;
                         DO LA=1 TO N1;

                         G4(LA,LB,LC,LD)=
                           G4(LA,LB,LC,LD)
                           +X1(LM)*G4(LA-1,LB,LC,LD)
                           +X2(LM)*G4(LA,LB-1,LC,LD)
                           +X3(LM)*G4(LA,LB,LC-1,LD)
                           +X4(LM)*G4(LA,LB,LC,LD-1);

                         END;
                    END;
               END;
          END;

     END;
     END FOURJOB;
```

```
PUTG:   PROC(N1,N2,N3,N4,GP);
   DCL GP(*,*,*,*) FLOAT BIN;
   DCL (N1,N2,N3,N4,NT) FIXED BIN;

   PUT SKIP(3)('G FOLLOWS:');
   PUT SKIP(2);
   NT=N1+N2+N3+N4;
   LCOUNT=3;

   DO IA=0 TO N1;
        DO IB=0 TO N2;
             DO IC=0 TO N3;
                  DO ID=0 TO N4;

                  IF LCOUNT=0 THEN DO;
                       PUT SKIP;
                       LCOUNT=3; END;
                  PUT  EDIT('  G(',IA,',',IB,',',IC,',',ID,
                       ')=',GP(IA,IB,IC,ID))
                   (A,F(N1/10+1),A,F(N2/10+1),A,F(N3/10+1),A,
                       F(N4/10+1),A,E(15,7));
                  LCOUNT=LCOUNT-1;

                  END;
             END;
        END;
   END;

   END PUTG;
FINISH: END NORM;
```

APPENDIX D: THRUPUT COMPUTATION USING G VALUES

```
TPUT:   PROC OPTIONS(MAIN);
   DCL (NA,NB,NC,ND) FIXED BIN;
   DCL MU(4) FLOAT BIN;
   DCL NCASE FIXED BIN;
   DCL (ZA,ZB,ZC,ZD) FLOAT BIN;

   /* AN INPUT FILE, GVALUES, MUST BE ALLOCATED. NOTE
   THAT IT CONTAINS THE PARAMETERS OF THE NETWORK
   (SEE FOLLOWING STATEMENT) AS WELL AS
   G COMPUTED FROM NORM.   */

   GET FILE(GVALUES) LIST(NA,NB,NC,ND,NCASE,ZA,ZB,ZC,ZD,MU);
   PUT SKIP(3) EDIT('VALUES OF G MATRIX ARE:', 'NA= ',NA,
        'NB= ',NB,'NC= ',NC,'ND= ',ND)
        (A,SKIP,(4)(A,F(2),X(3)));
   PUT SKIP(3) EDIT('NETWORK CHARACTERISTICS ARE:','CASE ',
        NCASE,'ZA= ',ZA,'ZB= ',ZB,'ZC= ',ZC,'ZD= ',ZD,
        'MU(1)= ',MU(1),'MU(2)= ',MU(2),
        'MU(3)= ',MU(3),'MU(4)= ',MU(4))
        (A,SKIP,A,F(2),X(3),(4)(A,F(6,2),X(3)),SKIP,
        (4)(A,E(15,7),X(2)));

   BEGIN;
   DCL (TSA,TSB,TSC,TSD,NV,TTOT) FLOAT BIN;
   DCL G(0:NA,0:NB,0:NC,0:ND) FLOAT BIN;
   DCL (VA,VB,VC,VD) FIXED BIN;
   DCL (LISTF,LOADF) CHAR(1);
   DCL RFLAG FIXED BIN;
   DCL (PA,PB,PC,PD) FLOAT BIN;
TEDIT:  FORMAT(A,X(3),(3)(A,F(2),X(3)),SKIP);
   GET FILE(GVALUES) LIST(G);
   PUT SKIP(3) LIST('G LISTING DESIRED? - Y OR N :');
   GET LIST(LISTF);
   IF LISTF='Y' THEN CALL PUTG(NA,NB,NC,ND,G);

   /* COMPUTE THE DESIRED THRUPUTS */
   /* COMPUTE SHORT CIRCUIT THRUPUT */

RESTART:
   PUT SKIP(3) LIST('TOTAL JOB VECTOR:');
   GET LIST(NA,NB,NC,ND);
   RFLAG=0;
   PUT SKIP(3) EDIT('JOB LOADING IS:',
        'NA= ',NA,'NB= ',NB,'NC= ',NC,'ND= ',ND)
        (A,SKIP,(4)(A,F(2),X(2)));
   TSA=G(NA-1,NB,NC,ND)/G(NA,NB,NC,ND);
   TSB=G(NA,NB-1,NC,ND)/G(NA,NB,NC,ND);
   TSC=G(NA,NB,NC-1,ND)/G(NA,NB,NC,ND);
```

```
    TSD=G(NA,NB,NC,ND-1)/G(NA,NB,NC,ND);
    TTOT=TSA+TSB+TSC+TSD;
    PUT SKIP(3) EDIT('SHORT CIRCUIT THRUPUT FOLLOWS:',
         ' TTOT= ',TTOT,' TSA= ',TSA,' TSB= ',TSB,' TSC= ',TSC,
         ' TSD= ',TSD)
         (A,SKIP,(5)(X(3),A,F(6,3)));

    /* COMPUTE BRANCHING RATIOS */

    PA=TSA/TTOT;
    PB=TSB/TTOT;
    PC=TSC/TTOT;
    PD=TSD/TTOT;
    PUT SKIP(3) EDIT('THRUPUT BRANCHING IS:',
         ' PA= ',PA,' PB= ',PB,' PC= ',PC,' PD= ',PD)
         (A,SKIP,(4)(X(3),A,F(6,3)));

    /*COMPUTE THE THRUPUT AS A FUNCTION OF THE
      NUMBER OF JOBS IN EACH JOB CLASS WITH THE
      OTHER JOBS HELD FIXED.*/

    RFLAG=1;
NEWLOAD:
    IF RFLAG=0 THEN DO;
         PUT SKIP(2) LIST
              ('INPUT THE ACTIVE VALUES OF JOB CLASSES:');
         PUT SKIP;
         GET LIST(VA,VB,VC,VD);
         END;
         ELSE DO; VA=NA;VB=NB;VC=NC;VD=ND;END;

    PUT SKIP(3) EDIT('THRUPUT VS. AJOBS',
         'NB= ',VB,'NC= ',VC,'ND= ',VD)(R(TEDIT));
    DO IA=1 TO VA;
         TSA=G(IA-1,NB,NC,ND)/G(IA,NB,NC,ND);
         PUT SKIP EDIT('NA= ',IA,'TSA= ',TSA)
              (A,F(2),X(3),A,F(7,3));
    END;
    PUT SKIP(3) EDIT('THRUPUT VS. BJOBS',
         'NA= ',VA,'NC= ',VC,'ND= ',VD)(R(TEDIT));
    DO IB=1 TO VB;
         TSB=G(NA,IB-1,NC,ND)/G(NA,IB,NC,ND);
         PUT SKIP EDIT('NB= ',IB,'TSB= ',TSB)
              (A,F(2),X(3),A,F(7,3));
    END;
    PUT SKIP(3) EDIT('THRUPUT VS. CJOBS',
         'NA= ',VA,'NB= ',VB,'ND= ',VD)(R(TEDIT));
    DO IC=1 TO VC;
         TSC=G(NA,NB,IC-1,ND)/G(NA,NB,IC,ND);
         PUT SKIP EDIT('NC= ',IC,'TSC= ',TSC)
              (A,F(2),X(3),A,F(7,3));
    END;
```

```
    PUT SKIP(3) EDIT('THRUPUT VS. DJOBS',
         'NA= ',VA,'NB= ',VB,'NC= ',VC)(R(TEDIT));
    DO ID=1 TO VD;
         TSD=G(NA,NB,NC,ID-1)/G(NA,NB,NC,ID);
         PUT SKIP EDIT('ND= ',ID,'TSD= ',TSD)
              (A,F(2),X(3),A,F(7,3));
    END;

    RFLAG=0;
    PUT PAGE;
    PUT SKIP(6) LIST
         ('NEW LOADING DESIRED? - Y,N OR R FOR RESTART:');
    GET LIST(LOADF);
    IF LOADF='Y' THEN GO TO NEWLOAD;
    IF LOADF='R' THEN GO TO RESTART;

    END; /* BEGIN BLOCK */

PUTG:   PROC(N1,N2,N3,N4,GP);
    DCL GP(*,*,*,*) FLOAT BIN;
    DCL (N1,N2,N3,N4,NT) FIXED BIN;

    PUT SKIP(3)('G FOLLOWS:');
    PUT SKIP(2);
    NT=N1+N2+N3+N4;
    LCOUNT=3;

    DO IA=0 TO N1;
         DO IB=0 TO N2;
              DO IC=0 TO N3;
                   DO ID=0 TO N4;

                   IF LCOUNT=0 THEN DO;
                        PUT SKIP;
                        LCOUNT=3; END;
                   PUT  EDIT('  G(',IA,',',IB,',',IC,',',ID,
                        ')=',GP(IA,IB,IC,ID))
                    (A,F(N1/10+1),A,F(N2/10+1),A,F(N3/10+1),A,
                        F(N4/10+1),A,E(15,7));
                   LCOUNT=LCOUNT-1;

                   END;
              END;
         END;
    END;

    END PUTG;

    END TPUT;
```

APPENDIX E: EXACT HNORM COMPUTATION FOR COMPUTER NETWORKS WITH MULTIPLE JOB TYPES

```
HNORM4:   PROC OPTIONS (MAIN);

   /* COMPUTES THE NORMALIZATION CONSTANTS FOR M NODES AND
   UP TO FOUR JOB CLASSES:
        A - 1ST JOB CLASS
        NA - NUMBER OF A JOBS
        B - 2ND JOB CLASS
        NB - NUMBER OF B JOBS
        C - 3RD JOB CLASS
        NC - NUMBER OF C JOBS
        D - 4TH JOB CLASS
        ND - NUMBER OF D JOBS

   H(NA,NB,NC,ND) IS AN ARRAY FOR THE LAST OR HIGHEST VALUE OF
        M AND IS MAINTAINED IN STORAGE. THE M IS IMPLIED.

   XA(M),XB(M),XC(M),XD(M) ARE THE RESPECTIVE, RELATIVE
        UTILIZATIONS FOR JOB CLASSES A,B,C,D.  */

   /*  INPUT IS FREEFORMAT:
        NO OF CLASSES, NCLASS;
        M, THE NUMBER OF NODES;
        NO OF INDIVIDUAL CLASSES, NA,NB,NC,ND;
        RELATIVE UTILIZATIONS, XA,XB,XC,XD.
             THESE ARE VECTORS OF LENGTH M;
        SERVICE RATES AT THE NODES (1,2,3,4);
        A FLAG FOR PRINTER OUTPUT OF THE H,  -Y OR N;


   A OUTPUT FILE, HDATA, MUST BE ALLOCATED FOR THE
             G MATRIX.    */

   DCL(LA,LB,LC,LD) FIXED BIN;
   DCL  (M,NCLASS,NA,NB,NC,ND) FIXED BIN ;
   DCL  (IA,IB,IC,ID) FIXED BIN;

   GET FILE(HDATA) LIST(NCLASS,M);
   PUT SKIP DATA(NCLASS,M);

   /* DETERMINE THE NUMBER OF CLASSES */

   IF NCLASS=1 THEN GO TO FINISH;
   IF NCLASS=2 THEN GO TO FINISH;
   IF NCLASS=3 THEN GO TO FINISH;
   IF NCLASS=4 THEN GO TO CLASS4;
```

```
   /* ESTABLISHED FOR 4 CLASSES.
   OTHERS ARE SUBCASES OF 4 AND ARE
   NOT INCLUDEED IN THIS LISTING.  */


CLASS4:      /* CASE OF FOUR JOB CLASSES */

   GET FILE(HDATA) LIST(NA,NB,NC,ND);
   PUT SKIP DATA(NA,NB,NC,ND);

   BEGIN;
   DCL (XA(M),XB(M),XC(M),XD(M)) FLOAT BIN;
   DCL (H(0:NA,0:NB,0:NC,0:ND), HC(0:NC,0:M), HD(0:ND,0:M),
    HA(0:NA,0:M), HB(0:NB,0:M)) FLOAT BIN;
   DCL (HAB(0:NA,0:NB,0:M),HAC(0:NA,0:NC,0:M),
        HBC(0:NB,0:NC,0:M)) FLOAT BIN;
   DCL (HAD(0:NA,0:ND,0:M),HBD(0:NB,0:ND,0:M),
        HCD(0:NC,0:ND,0:M)) FLOAT BIN;
   DCL (HABC(0:NA,0:NB,0:NC,0:M),
        HABD(0:NA,0:NB,0:ND,0:M),
        HACD(0:NA,0:NC,0:ND,0:M),
        HBCD(0:NB,0:NC,0:ND,0:M)) FLOAT BIN;
   DCL (U(M),UA(M),UB(M),UC(M),UD(M)) FLOAT BIN;
   DCL (T(M),TA(M),TB(M),TC(M),TD(M)) FLOAT BIN;
   DCL MU(M) FLOAT BIN;
   DCL (ZA,ZB,ZC,ZD) FLOAT BIN;
   DCL (TRA,TRB,TRC,TRD,TR) FLOAT BIN;
   DCL (PA,PB,PC,PD) FLOAT BIN;
   DCL (MNA,MNB,MNC,MND) FLOAT BIN;
   DCL (VA,VB,VC,VD) FLOAT BIN;
   DCL (MTPA,MTPB,MTPC,MTPD,MTPTOT) FLOAT BIN;
   DCL PFLAG CHAR(1);
HEDIT:  FORMAT(A,F(NA/10+1),A,F(NB/10+1),A,F(NC/10+1),
        A,F(ND/10+1),A,E(15,8),SKIP);
MEDIT:  FORMAT(A,SKIP,(4)(X(3),A,F(6,3)));
   GET FILE(HDATA) LIST(XA,XB,XC,XD);
   PUT SKIP DATA(XA,XB,XC,XD);
   GET FILE(HDATA) LIST(MU);
   PUT SKIP DATA(MU);
   GET FILE(HDATA) LIST(ZA,ZB,ZC,ZD);
   PUT SKIP DATA(ZA,ZB,ZC,ZD);
   IF NA=0 THEN PUT SKIP LIST('CLASS ERROR');

   /* INITIALIZE THE H */
   H=1.0; HA=1.0; HB=1.0; HC=1.0; HD=1.0;
   HAB=1.0; HAC=1.0;HBC=1.0;
   HAD=1.0; HBD=1.0; HCD=1.0;
   HABC=1.0; HABD=1.0;HACD=1.0;HBCD=1.0;
```

```
CALL ONEJOB(NA,XA,HA,ZA);
CALL ONEJOB(NB,XB,HB,ZB);
CALL ONEJOB(NC,XC,HC,ZC);
CALL ONEJOB(ND,XD,HD,ZD);

CALL TWOJOB(NA,NB,XA,XB,HAB,HB,HA,ZA,ZB);
CALL TWOJOB(NA,NC,XA,XC,HAC,HC,HA,ZA,ZC);
CALL TWOJOB(NB,NC,XB,XC,HBC,HC,HB,ZB,ZC);
CALL TWOJOB(NA,ND,XA,XD,HAD,HD,HA,ZA,ZD);
CALL TWOJOB(NB,ND,XB,XD,HBD,HD,HB,ZB,ZD);
CALL TWOJOB(NC,ND,XC,XD,HCD,HD,HC,ZC,ZD);

CALL THREJOB(NA,NB,NC,XA,XB,XC,HABC,HBC,HAC,HAB,ZA,ZB,ZC);
CALL THREJOB(NA,NB,ND,XA,XB,XD,HABD,HBD,HAD,HAB,ZA,ZB,ZD);
CALL THREJOB(NA,NC,ND,XA,XC,XD,HACD,HCD,HAD,HAC,ZA,ZC,ZD);
CALL THREJOB(NB,NC,ND,XB,XC,XD,HBCD,HCD,HBD,HBC,ZB,ZC,ZD);

/* COMPUTE THE H */

CALL FOURJOB(NA,NB,NC,ND,XA,XB,XC,XD,
     H,HBCD,HACD,HABD,HABC,ZA,ZB,ZC,ZD);
PUT SKIP(2) LIST('PRINTOUT OF H DESIRED? - Y, N, F OR B ');
GET LIST(PFLAG);
IF (PFLAG='Y'   PFLAG='B') THEN CALL PUTH(NA,NB,NC,ND,H);
IF (PFLAG='B'   PFLAG='F') THEN DO;
     PUT FILE(H4OUT) LIST(NA,NB,NC,ND);
     PUT SKIP(2) LIST('WHAT IS THE CASE?');
     GET LIST(NCASE);
     PUT FILE(H4OUT) LIST(NCASE);
     PUT FILE(H4OUT) LIST(ZA,ZB,ZC,ZD);
     PUT FILE(H4OUT) LIST(MU);
     PUT FILE(H4OUT) EDIT(H)
          ((4)(E(15,8),X(3)),SKIP);
     END; /* THEN DO */
PUT SKIP(2) EDIT('H(',NA-1,',',NB,',',NC,',',ND,')= ',
     H(NA-1,NB,NC,ND))(R(HEDIT));
PUT SKIP(2) EDIT('H(',NA,',',NB-1,',',NC,',',ND,')= ',
     H(NA,NB-1,NC,ND))(R(HEDIT));
PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC-1,',',ND,')= ',
     H(NA,NB,NC-1,ND))(R(HEDIT));
PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC,',',ND-1,')= ',
     H(NA,NB,NC,ND-1))(R(HEDIT));
PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC,',',ND,')= ',
     H(NA,NB,NC,ND))(R(HEDIT));

/* COMPUTE MEAN THRUPUT */

VA=NA; VB=NB; VC=NC; VD=ND;
MTPA=H(NA-1,NB,NC,ND)/H(NA,NB,NC,ND)*VA/ZA;
MTPB=H(NA,NB-1,NC,ND)/H(NA,NB,NC,ND)*VB/ZB;
MTPC=H(NA,NB,NC-1,ND)/H(NA,NB,NC,ND)*VC/ZC;
MTPD=H(NA,NB,NC,ND-1)/H(NA,NB,NC,ND)*VD/ZD;
```

```
      PUT SKIP(3) EDIT('SHORT CIRCUIT THRUPUT FOLLOWS:',
           ' MTPA= ',MTPA,' MTPB= ',MTPB,' MTPC= ',MTPC,
           ' MTPD= ',MTPD)(R(MEDIT));


      /* JOB PROBABILITY FOLLOWS */

      MTPTOT=MTPA+MTPB+MTPC+MTPD;
      PA=MTPA/MTPTOT;
      PB=MTPB/MTPTOT;
      PC=MTPC/MTPTOT;
      PD=MTPD/MTPTOT;
      PUT SKIP(3) EDIT('JOB PROBABILITIES ARE:','PA= ',PA,
           'PB= ',PB,'PC= ',PC,'PD= ',PD)(R(MEDIT));

      /* MEAN QUEUE LENGTHS FOLLOW */

      HNORM=H(NA,NB,NC,ND);
      MNA=VA*(1.0-(H(NA-1,NB,NC,ND)/HNORM));
      MNB=VB*(1.0-(H(NA,NB-1,NC,ND)/HNORM));
      MNC=VC*(1.0-(H(NA,NB,NC-1,ND)/HNORM));
      MND=VD*(1.0-(H(NA,NB,NC,ND-1)/HNORM));
      PUT SKIP(3) EDIT('MEAN QUEUE LENGTHS ARE:',
           'MNA= ',MNA,'MNB= ',MNB,'MNC= ',MNC,'MND= ',MND)
           (R(MEDIT));

      /* RESPONSE TIMES */

      TRA=MNA/MTPA;
      TRB=MNB/MTPB;
      TRC=MNC/MTPC;
      TRD=MND/MTPD;
      TR = TRA*PA + TRB*PB + TRC*PC + TRD*PD;
      PUT SKIP(3) EDIT('MEAN RESPONSE TIMES ARE:',
           'TRA= ',TRA,'TRB= ',TRB,'TRC= ',TRC,'TRD= ',TRD)
           (R(MEDIT));
      PUT SKIP EDIT('TR= ',TR)(X(3),A,F(6,3));

      END; /* BEGIN BLOCK CLASS3 */
      GO TO FINISH;

   ONEJOB: PROC(N1,X1,H1,Z1);
      DCL N1 FIXED BIN;
      DCL X1(*) FLOAT BIN;
      DCL Z1 FLOAT BIN;
      DCL H1(*,*) FLOAT BIN;

      H1(*,0)=1.0;
      H1(0,*)=1.0;
      H1(0,0)=1.0;
```

```
    DO LM=1 TO M;
         DO LA=1 TO N1;
              H1(LA,LM)=H1(LA,LM-1) + X1(LM) *LA/Z1*H1(LA-1,LM);
         END;
    END;
    END ONEJOB;


 TWOJOB: PROC(N1,N2,X1,X2,H2,H2Z1,H2Z2,Z1,Z2);
    DCL (N1,N2) FIXED BIN;
    DCL (X1(*),X2(*)) FLOAT BIN;
    DCL (Z1,Z2) FLOAT BIN;
    DCL (H2(*,*,*),H2Z1(*,*),H2Z2(*,*)) FLOAT BIN;
    DO LM=1 TO M;
         H2(*,0,LM)=H2Z2(*,LM);
         H2(0,*,LM)=H2Z1(*,LM);
         DO LB=1 TO N2;
              DO LA=1 TO N1;
                   H2(LA,LB,LM) = H2(LA,LB,LM-1) +
                      X1(LM)*H2(LA-1,LB,LM)*LA/Z1
                    + X2(LM)*H2(LA,LB-1,LM)*LB/Z2;
              END;
         END;
    END;
    END TWOJOB;

 THREJOB:    PROC(N1,N2,N3,X1,X2,X3,H3,H3Z1,H3Z2,H3Z3,Z1,Z2,Z3);
    DCL (N1,N2,N3) FIXED BIN;
    DCL (X1(*),X2(*),X3(*)) FLOAT BIN;
    DCL (H3(*,*,*,*),H3Z1(*,*,*),
         H3Z2(*,*,*),H3Z3(*,*,*)) FLOAT BIN;
    DCL (Z1,Z2,Z3) FLOAT BIN;

    DO LM=1 TO M;
         H3(0,*,*,LM)=H3Z1(*,*,LM);
         H3(*,0,*,LM)=H3Z2(*,*,LM);
         H3(*,*,0,LM)=H3Z3(*,*,LM);

         DO LC=1 TO N3;
              DO LB=1 TO N2;
                   DO IA=1 TO N1;
                   H3(IA,LB,LC,LM)=H3(LA,LB,LC,LM-1)
                     +X1(LM)*H3(LA-1,LB,LC,LM)*LA/Z1
                     +X2(LM)*H3(LA,LB-1,LC,LM)*LB/Z2
                     +X3(LM)*H3(LA,LB,LC-1,LM)*LC/Z3;

                   END;
              END;
         END;
    END;
    END THREJOB;
```

```
FOURJOB:      PROC(N1,N2,N3,N4,X1,X2,X3,X4,
        H4,H4Z1,H4Z2,H4Z3,H4Z4,Z1,Z2,Z3,Z4);
   DCL(N1,N2,N3,N4) FIXED BIN;
   DCL(X1(*),X2(*),X3(*),X4(*)) FLOAT BIN;
   DCL(H4(*,*,*,*),H4Z1(*,*,*,*),H4Z2(*,*,*,*),
        H4Z3(*,*,*,*),H4Z4(*,*,*,*)) FLOAT BIN;
   DCL (Z1,Z2,Z3,Z4) FLOAT BIN;

   DO LM=1 TO M;
        H4(0,*,*,*)=H4Z1(*,*,*,LM);
        H4(*,0,*,*)=H4Z2(*,*,*,LM);
        H4(*,*,0,*)=H4Z3(*,*,*,LM);
        H4(*,*,*,0)=H4Z4(*,*,*,LM);

        DO LD=1 TO N4;
             DO LC=1 TO N3;
                  DO LB=1 TO N2;
                       DO LA=1 TO N1;

                       H4(LA,LB,LC,LD)=
                         H4(LA,LB,LC,LD)
                         +X1(LM)*H4(LA-1,LB,LC,LD)*LA/Z1
                         +X2(LM)*H4(LA,LB-1,LC,LD)*LB/Z2
                         +X3(LM)*H4(LA,LB,LC-1,LD)*LC/Z3
                         +X4(LM)*H4(LA,LB,LC,LD-1)*LD/Z4;

                       END;
                  END;
             END;
        END;

   END;
   END FOURJOB;

PUTH:   PROC(N1,N2,N3,N4,HP);
   DCL HP(*,*,*,*) FLOAT BIN;
   DCL (N1,N2,N3,N4,NT) FIXED BIN;

   PUT SKIP(3)('H FOLLOWS:');
   PUT SKIP(2);
   NT=N1+N2+N3+N4;
   LCOUNT=3;

   DO IA=0 TO N1;
        DO IB=0 TO N2;
             DO IC=0 TO N3;
                  DO ID=0 TO N4;

                  IF LCOUNT=0 THEN DO;
                       PUT SKIP;
                       LCOUNT=3; END;
```

```
                    PUT  EDIT('  H(',IA,',',IB,',',IC,',',ID,
                         ')=',HP(IA,IB,IC,ID),';')
                     (A,F(N1/10+1),A,F(N2/10+1),A,F(N3/10+1),A,
                         F(N4/10+1),A,E(13,6),A);
                    LCOUNT=LCOUNT-1;

                    END;
               END;
          END;
     END;

     END PUTH;
FINISH: END HNORM4;
```

APPENDIX F: HNCRM MODEL RESPONSE TIME COMPUTATION

```
HRTIME: PROC OPTIONS(MAIN);

   DCL H(0:NA,0:NB,0:NC,0:ND) FLOAT BIN CTL;
   DCL NCASE FIXED BIN;
   DCL (ZA,ZB,ZC,ZD) FLOAT BIN;
   DCL (NA,NB,NC,ND) FIXED BIN;
   DCL HNORM FLOAT BIN;
   DCL (TRA,TRB,TRC,TRD,TR) FLOAT BIN;
   DCL (MNB,MNC,MND,MNA) FLOAT BIN;
   DCL (MTPA,MTPB,MTPC,MTPD,MTPTOT) FLOAT BIN;
   DCL MU(4) FLOAT BIN;
HEDIT:  FORMAT(A,F(NA/10+1),A,F(NB/10+1),A,F(NC/10+1),
        A,F(ND/10+1),A,E(15,8),SKIP);
MEDIT:  FORMAT(A,SKIP,(4)(X(3),A,F(6,3)));
   DCL PFLAG CHAR(1);

   /* MUST ALLOCATE AN INPUT FILE, HVALUES,
   CONTAINING THE NETWORK PARAMETERS AND H. */

   GET FILE(HVALUES) LIST(NA,NB,NC,ND,NCASE,ZA,ZB,ZC,ZD,MU);
   ALLOCATE H;
   GET FILE(HVALUES) LIST(H);
   PUT SKIP(3) EDIT('VALUES OF H MATRIX ARE:', 'NA= ',NA,
        'NB= ',NB,'NC= ',NC,'ND= ',ND)
        (A,SKIP,(4)(A,F(2),X(3)));
   PUT SKIP(3) EDIT('NETWORK CHARACTERISTICS ARE:','CASE ',
        NCASE,'ZA= ',ZA,'ZB= ',ZB,'ZC= ',ZC,'ZD= ',ZD,
        'MU(1)= ',MU(1),'MU(2)= ',MU(2),
        'MU(3)= ',MU(3),'MU(4)= ',MU(4))
        (A,SKIP,A,F(2),X(3),(4)(A,F(6,2),X(3)),SKIP,
        (4)(A,E(15,7),X(2)));

   PUT SKIP(3) LIST('INPUT THE LOAD VECTOR:');
   GET LIST(NA,NB,NC,ND);

   PUT SKIP(2) LIST('PRINTOUT OF H DESIRED? - Y OR N:');
   GET LIST(PFLAG);
   IF (PFLAG='Y') THEN CALL PUTH(NA,NB,NC,ND,H);

RESTART:
   PUT SKIP(2) EDIT('H(',NA-1,',',NB,',',NC,',',ND,')= ',
        H(NA-1,NB,NC,ND))(R(HEDIT));
   PUT SKIP(2) EDIT('H(',NA,',',NB-1,',',NC,',',ND,')= ',
        H(NA,NB-1,NC,ND))(R(HEDIT));
   PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC-1,',',ND,')= ',
        H(NA,NB,NC-1,ND))(R(HEDIT));
   PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC,',',ND-1,')= ',
        H(NA,NB,NC,ND-1))(R(HEDIT));
```

```
PUT SKIP(2) EDIT('H(',NA,',',NB,',',NC,',',ND,')= ',
     H(NA,NB,NC,ND))(R(MEDIT));
/* COMPUTE MEAN THRUPUT */


MTPA=H(NA-1,NB,NC,ND)/H(NA,NB,NC,ND)*NA/ZA;
MTPB=H(NA,NB-1,NC,ND)/H(NA,NB,NC,ND)*NB/ZB;
MTPC=H(NA,NB,NC-1,ND)/H(NA,NB,NC,ND)*NC/ZC;
MTPD=H(NA,NB,NC,ND-1)/H(NA,NB,NC,ND)*ND/ZD;
PUT SKIP(3) EDIT('SHORT CIRCUIT THRUPUT FOLLOWS:',
     ' MTPA= ',MTPA,' MTPB= ',MTPB,' MTPC= ',MTPC,
     ' MTPD= ',MTPD)(R(MEDIT));

/* JOB PROBABILITY FOLLOWS */

MTPTOT=MTPA+MTPB+MTPC+MTPD;
PA=MTPA/MTPTOT;
PB=MTPB/MTPTOT;
PC=MTPC/MTPTOT;
PD=MTPD/MTPTOT;
PUT SKIP(3) EDIT('JOB PROBABILITIES ARE:','PA= ',PA,
     'PB= ',PB,'PC= ',PC,'PD= ',PD)(R(MEDIT));

/* MEAN QUEUE LENGTHS FOLLOW */

HNORM=H(NA,NB,NC,ND);
MNA=NA*(1.0-(H(NA-1,NB,NC,ND)/HNORM));
MNB=NB*(1.0-(H(NA,NB-1,NC,ND)/HNORM));
MNC=NC*(1.0-(H(NA,NB,NC-1,ND)/HNORM));
MND=ND*(1.0-(H(NA,NB,NC,ND-1)/HNORM));
PUT SKIP(3) EDIT('MEAN QUEUE LENGTHS ARE:',
     'MNA= ',MNA,'MNB= ',MNB,'MNC= ',MNC,'MND= ',MND)
     (R(MEDIT));

/* RESPONSE TIMES */

TRA=MNA/MTPA;
TRB=MNB/MTPB;
TRC=MNC/MTPC;
TRD=MND/MTPD;
TR = TRA*PA + TRB*PB + TRC*PC + TRD*PD;
PUT SKIP(3) EDIT('MEAN RESPONSE TIMES ARE:',
     'TRA= ',TRA,'TRB= ',TRB,'TRC= ',TRC,'TRD= ',TRD)
     (R(MEDIT));
PUT SKIP EDIT('TR= ',TR)(X(3),A,F(6,3));


PUT PAGE;
PUT SKIP(6) LIST('NEW LOAD VECTOR IS:');
GET LIST(NA,NB,NC,ND);
IF NA<100 THEN GO TO RESTART;
```

```
PUTH:   PROC(N1,N2,N3,N4,HP);
   DCL HP(*,*,*,*) FLOAT BIN;
   DCL (N1,N2,N3,N4,NT) FIXED BIN;

   PUT SKIP(3)('H FOLLOWS:');
   PUT SKIP(2);
   NT=N1+N2+N3+N4;
   LCOUNT=3;

   DO IA=0 TO N1;
        DO IB=0 TO N2;
             DO IC=0 TO N3;
                  DO ID=0 TO N4;

                  IF LCOUNT=0 THEN DO;
                       PUT SKIP;
                       LCOUNT=3; END;
                  PUT  EDIT('  H(',IA,',',IB,',',IC,',',ID,
                       ')=',HP(IA,IB,IC,ID),';')
                   (A,F(N1/10+1),A,F(N2/10+1),A,F(N3/10+1),A,
                       F(N4/10+1),A,E(13,6),A);
                  LCOUNT=LCOUNT-1;

                  END;
             END;
        END;
   END;

   END PUTH;
   END HRTIME;
```

REFERENCES

Abramson, Norman and Kuo, Franklin. Computer Communication Networks, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1973.

ACM (Association for Computing Machinery). "Proceedings of the Third Workshop on Computer Architecture for Non-Numeric Procesing." University of Syracuse, Syracuse, New York: n.p., May 17-18, 1977.

Arden, B. W. and Berenbaum. "A Mulit-Microprocessor Computer System Architecture." Proceedings of the Fifth Symposium on Operating Systems Principles, ACM Vol. 9, No. 5, (19-21 November, 1975).

Aschim, Frode. "Data Base Networks - An Overview." ISDOS Working Paper No. 79. Ann Arbor, Michigan: The University of Michagan, 1973.

Ashenhurst, R. L., and Vonderohe, R. H. "A Hierarchical Network." Datamation. February 19, 1975, pp. 40-44.

Asplund, C. "ASQ User's Manual." Department of Computer Science, Duke University, June 1976.

Aupperle, Eric M. "Merit Network Re-Examined." IEEE COMPCON, 1973, pp. 25-29.

Ballard, B. and Sigmon, T. Computer Science Department, Duke University. Conversation and PI/I Program. February, 1977.

Baskett, F., Chandy, K., Muntz, R. and Polacios, F. "Open, Closed, and Mixed Networks of Queues with Different Classes of Customers." JACM, Vol. 22, No. 2, April, 1975, pp. 248-260.

Baskett, F. and Muntz, R. "Networks of Queues", Proceedings of the 7th Annual Princeton Conference on Information Sciences and Systems, March 1973, pp. 428-434.

Baum, R. I. and Hsiao, D. K. "A Data Science Corporation Architecture." IEEE COMPCON, Spring, 1976, pp. 114-117.

Belford, Geneva G., Ray, John D., Sluiger, Suzanne and Wilcox, David A. Initial Mathematical Model Report. JTSA Document No. 5511, August 20, 1975.

Bhandiwad, R. A. and Williams, A. C. "Queueing Network Models of Computer Systems." IEEE 1974 Texas Conference on Computing Systems: 7-1-1 to 7-1-8.

Bowra, J. W. and Torng, H. C. "The Modeling and Design of Multiple Function-Unit Processors." IEEE Transactiong on Computers, Vol. c-25, No. 3, March 1976.

Bredt, T. H. "Analysis of Parallel Systems." IEEE Transactions on Computers, Volume c-20, November 1971, pp. 1403-1407.

Bressler et al., "A Multiprocessor For Communications Networks." undated memorandum, Bolt, Beranek & Newman, Inc.

Browne, J. C., Chandy, K. M., Brown,, R. M., Keller, T. W., Towsley, D. F. and Dirsley, C. W. "Hierarchical Techniques for the Development of Realistic Models of Complex Computer Systems." Proceedings of the IEEE, Vol. 63, No. 6, June 1975.

Burke, P. J. "The Output of a Queueing System." Operations Research, 4 (1956): 699-704.

Buzen, Jeffrey P. "Queueing Network Models of Multiprogramming." Ph.D. dissertation, Div. of Eng and Applied Physics, Harvard University, August 1971.

Buzen, J. P. "Computational Algorithms for Closed Queueing Networks with Exponential Servers." CACM, 16, No. 9, September 1973, pp. 527-531.

Buzen, J. P. "Cost Effective Analytic Tools for Computer Performance Evaluation." Proceedings COMPCON 75 (Eleventh IEEE Computer Society Conference), Washington, D. C., September 1975, pp. 293-296.

Buzen, J. P. "Fundamental Laws of Computer System Performance." Proceedings AFIP-ACM Sigmetrics International Symposium on Computer System Modeling, Measurement and Evaluation, Cambridge, Massachusetts, March 1976, pp. 200-210.

Buzen, Jeffrey P. "Operational Analysis: The Key to the New Generation of Performance Prediction Tools." Proceedings COMPCON 76, Washington, D. C., September 1976.

Canaday, R. H. et al., "A Back-End Computer for Data Base Management." CACM, Vol. 17, No. 10, October 1974.

Chandy, K. M. "The Analysis and Solutions for General Queueing Networks." Proceedings of the 6th Annual Princeton Conference on Information Sciences and Systems, March 1972, pp. 224-228.

Chandy, K. M. et al., "Design Automation and Queueing Networks." Proceedings of the 9th Annual Design Automation Workshop, Dallas, Texas, June 1972.

Chandy, K. M., Herzog, U., and Woo, L. "Approximate Analysis of General Queueing Networks." IBM Journal of Research and Development, Vol. 19, No. 1 (January 1975): pp. 43-49.

Chandy, K. M., Herzog, U. and Woo, L. "Parametric Analysis of Queueing Network Models." IBM Journal of Research and Development, Vol. 19, No. 3 (1975): pp. 36-47.

Chandy, K. M. and Sauer, C. H. "Approximate Analysis of Central Server Models." IBM Journal Research Division (May 1975): pp. 301-313.

Chou, Wushow and McGregor, Patrick V. "A Unified Simulation Model for Communication Processors." IEEE 1975 Computer Networks, Trends and Applications, pp. 40-46.

Coffman, Edward G. and Denning, Peter J. Operating Systems Theory. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1973.

Cohen, E. and Jefferson, D. "Protection in the Hydra Operating System." ACM Operating Systems Review, Vol. 9, No. 5, 19-21, November 1975, pp. 141-161.

Colella, A. M., O'Sullivan, M. J. and Carlino, D. J. Systems Simulation. Lexington, Massachusetts: Lexington Books, 1974.

Command and Control Technical Center, Joint Chiefs of Staff, Department of Defense. "Overview of the PWIN Operational Experiment Program." February 18, 1976.

Conte, S. D. Elementary Numerical Analysis. New York: McGraw-Hill, Inc., 1965.

Cosell, B. P. et. al. "An Operational System For Computer Resources Sharing." ACM Operating Systems Review, Vol. 9, No. 5, 19-21, November 1975.

Courtois, P. J. "Decomposability Instabilities and Saturation in Multiprogramming Systems." CACM, Vol. 18, No. 7, 1975, pp. 371-376.

Denning, P. J. "Understanding Performance Evaluation." Technology Transfer Inc. Seminar, Arlington, Virginia, 8-10 March 1977.

AD-A045 721 DUKE UNIV DURHAM N C DEPT OF ELECTRICAL ENGINEERING F/G 9/2
EQN MODELS FOR THE ANALYSIS AND DESIGN OF A COMPUTER NETWORK OF--ETC(U)
JUL 77 R L LEECH DAAG29-76-G-0309

UNCLASSIFIED ARO-14533.1-A-EL NL

3 OF 3
ADA
045721

END
DATE FILMED
11-77
DDC

Denning, P. J. and Muntz, R. R. "Queueing Theoretic Models." Computer Performance Evaluation: Report of the 1973 NBS/ACM Workshop, pp. 119-121.

Deo, Narsingh. Graph Theory with Applications to Engineering and Computer Science. Englewcod Cliffs, New Jersey: Prentice-Hill, Inc., 1977.

Doll, D. R. "Computer Networking - the Giant Step in Data Communications." Electronics Deskbcok, McGraw-Hill, 1973.

Drummond, M. E., Jr. Evaluation and Measurement Techniques for Computer Systems. Englewood Cliffs, New Jersey: Prentice Hall, Inc., 1973.

Enslow, P., ed. Multiprocessors and Parallel Processing. New York: John Wiley and Sons, 1974.

Farber, D. J. "A Ring Network." Datamation, February 1975, pp. 44-46.

Farber, David J. "Distributed Data Bases - An Exploration." IEEE Computer Networks, 1975, pp. 25-27.

Flynn, M. J. "Some Computer Organizations and Their Effectiveness." IEEE Transactions on Computers, Vol. c-21, September 1972, pp. 948-960.

Frank, H. "Computer Networks: Art to Science to Art." Networks, Vol 5, No. 1 (January 1945): 7-32.

Fraser, A. G. "A Virtual Channel Network." Datamation, February 1976, pp. 51-56.

Freeman, David N. "IBM and Multiprocessing." Datamation, March 1977, pp. 92-109.

Freeman, P. Software Systems and Principles, A Survey. Chicago, Illinois: Science Research Associates, Inc., 1975.

Fuller, Samuel H. "Performance Evaluation." In Introduction to Computer Architecture, pp. 474-541. Edited by Harold S. Stone. Chicago, Illinois: SRA Publishers, 1975.

Fuller, Samuel H. Colloquium presented at Duke University, November 1976.

Gallie, T. and Ramm, D. Computer Science/I: An Introduction to Structured Programming. Dubuque, Iowa: Kendall/Hunt Publishing Company, 1976.

Gerla, Mario. "Approximations and Bounds for the Topological Design of Distributed Computer Networks." IEEE 1975 Data Communications, pp. 4-9.

Giammo, Thomas. "Validation of a Computer Performance Model of the Exponential Queueing Network Family." Proceedings IFIP-ACM Sigmetrics International Symposium on Computer Performance Evaluation, Cambridge, Massachusetts, March 1976, pp. 44-58.

Goertzel, H. Department of Defense, Joint Chiefs of Staff, Worldwide Military Command and Control System, ADP Management Division. Conversation. 8 March 1976.

Gordon, W. J. and Newell, G. F. "Closed Queueing Systems with Exponential Servers." Operations Research 15, 2 (April 1967): pp. 254-265.

Green, P. E. and Tang, D. T. "Some Recent Developments in Teleprocessing System Optimization." IEEE Intercon Computers and Information Networks. March 26-30, 1973, p. 14/1.

Hakozaki, K., Yamamoto, Cno T., and Umemura, M. "Design and Evaluation System for Computer Architecture." Proceedings of the NCC, Vol. 42, 1973, pp. 81-86.

Hansen, P. B. Operating System Principles. Englewood Cliffs, New Jersey: Prentice-Hall, 1973.

Hellerman, H. and Conroy, T. F. Computer System Performance. New York: McGraw-Hill, 1975.

Herzog, Bertran. "Organizational Issues and the Computer Network Market." IEEE CCMPCON, 1973, pp. 11-14.

Hoare, C. A. R. "Monitors: An Operating System Structuring Concept." CACM, Vol. 17, No. 10, October 1974.

Hogarth, J. "Optimization and Analysis of Queueing Networks." Ph.D. dissertation, Department of Computer Science, University of Texas at Austin, 1975.

IBM Corporation. General Purpose Simulation System V User's Manual. White Plains, New York, 1971.

IEEE Computer Society and Association for Computing Machinery. Proceedings of the 4th Annual Symposium on Computer Architecture. Silver Spring, Maryland: n.p., March 23-25, 1977.

Information Research Associates. "User's Manual for the ASQ (Analytic Solution of Queues) System." 1975.

Jackson, James R. "Networks of Waiting Lines." Operation Research 5 (1957): pp. 518-521.

Jones, A., Siewiorek, D. and Fuller, S. "CM Project-Plans and Objectives." Working Memorandum Carnegie-Mellon, November 14, 1975.

Joseph, Earl O. "Distributed Function Computer Systems: Innovative Trends." IEEE COMPCON. Spring 1974, pp. 71-73.

Karp, R. M. and Miller, R. E. "Parallel Program Schemata." Journal of Computer and System Sciences 3 (1969): pp. 147-195.

Katzan, H. Computer Operating Systems. New York: Van Nostrand Company, 1973.

Kay, Ira M., Kinko, T. M., and Van Houweling, D. E. "GPSS/SIMSCRIPT - The Dominant Simulation Languages." IEEE 1975 Simulation Conference, pp. 141-154.

Keller, T. W., Towsley, D. F., Chandy, K. M. [illegible] Browne, J. C. "A Tool For Network Design: The [illegible]atic Analysis of Stochastic Models of Computer Netw[illegible]." IEEE 1973 Computer Society International Conference, pp. 151-153.

Kleinrock, L. Communication Nets: Stochastic Message Flow and Delay. New York: Dover, 1972.

Kleinrock, L. "Sequential Processing Machines (SPM) analyzed with a Queueing Theory Model." JACM, 13, 2 (April 1966): pp. 179-193.

Kleinrock, L. Analytical and Simulation Methods in Computer Network Design." AFIPS Conference Proceedings, Vol. 36, 1970, pp. 569-579.

Kleinrock, Leonard. Queueing Systems. Volume I: Theory. New York: John Wiley and Sons, 1975.

Kleinrock, Leonard. Queueing Systems. Volume II: Computer Applications. New York: John Wiley and Sons, 1975.

Kleinrock, L. and Naylor, W. E. "On Measured Behavior of the ARPA Network." AFIPS Conference Proceedings 43, May 1974, pp. 767-780.

Kleinrock, Leonard and Opderbeck, Holger. "Throughput in the ARPANET - Protocol and Measurement." IEEE 1975 Data Communications, pp. 6-1, 6-11.

Lavia, Anthony and Manning, Eric. "Pertubation Techniques for Topological Optimization of Computer Networks." IEEE 1975 Data Communications, pp. 4-16, 4-23.

Leech, R. L. "The Design and Analysis of a Network of Functionalized Processors." Project Report, Department of Computer Science, Duke University, October 21, 1976.

Levin, R. et. al. "Policy/Mechanism Separation in Hydra." ACM Operating Systems Review, Vol. 9, No. 5, 19-21, November 1965, pp. 132-141.

Little, J. D. C. "A Proof for the Queueing Formula L = lambda W." Operations Research 9 (1961): pp. 383-387.

Lucas, H. C. "Performance Evaluation and Monitoring." Computing Surveys, Vol. 3, No. 3, September 1971, pp. 79-91.

Madnick, J. J. and Donovan, S. E. Operating Systems. New York: McGraw-Hill Book Company, 1974.

Maekaura, M. and Boyd, D. L. "Two Models of Task Overlap Within Jobs of Multiprocessing Multiprogramming Systems." Proceedings of the 1976 International Conference on Parallel Processing. 1976, pp. 83-92.

Marill, T. and Stern, D. "The Datacomputer - A Network Data Utility." AFIPS Conference Proceedings National Computer Conference, 1975, pp. 389-395.

Maryanski, F. et. al. Usability and Feasibility of Back-End Minicomputers. Technical Documentary Report, U. S. Army Computer Systems Command, June 1975.

Moore, C. "Network Model for Large Scale Time Sharing Systems." Ph.D. dissertation, University of Michigan, 1971.

Mullery, A. P. The Distributed Control of Multiple Copies of Data, IBM Research Report Number 5782, December 29, 1975.

Muntz, R. R. "Scheduling and Resource Allocation in Computer Systems." In Software Systems Principles, A Survey, pp. 262-305, Edited by Peter Freeman, Chicago: SRA Publishers, Inc., 1975.

Muntz, R. R. and Wong, J. "Asymptotic Properties of Closed Queueing Network Models." Proceedings of the Eighth Annual Princeton Conference on Information Sciences and Systems, Princeton University, March 1974.

Nutt, Gary J. "Tutorial: Computer System Monitors," IEEE Computer, November 1975, pp. 51-61.

Ornstein, S. M., Stucki, M. J. and Clark, W. A. "A Functional Description of Macromodules." SJCC, 1967, pp. 337-355.

Parzen, Emanual. Stochastic Processes. San Francisco: Holden-Day, Inc., 1962.

Raphael, H. A. "Distributed Intelligence Microcomputer Design." IEEE COMPCON, February 1975, pp. 21-26.

Ravindran, V. K. and Thomas, T. "Characterization of Multiple Microprocessor Networks." IEEE COMPCON, March 1973, pp. 133-137.

Reiser, M. "Intractive Modeling of Computer Systems." IBM Systems Journal 4 (1976): pp. 309-327.

Roberts, Lawrence G. "Network Rationale: 5-Year Re-Evaluation." IEEE COMPCON 73, pp. 3-5.

Rose, C. A. "Validation of A Queueing Model with Classes of Customers." Proceddings AFIP-ACM Sigmetrics International Symposium of Computer System Modeling, Measurement and Evaluation, Cambridge, Massachusetts, March 1976, pp. 200-210.

Schriber, T. J. Simulation Using GPSS. New York: John Wiley and Sons, 1974.

Shu, Nan C. and Lum, Vincent Y. An Approach to Data Migration in Computer Networks: IBM Research Report RJ17013, Jauary 1976.

Stone, H. ed. Introduction to Computer Architecture. Chicago: Science Research Associates, 1975.

Trivedi, Kishor S. Computer Science Department, Duke University. Conversation. July 19, 1976.

Trivedi, Kishor S. "On the Design and Use of High Performance Computing Systems." Proceedings IMACS-AICA-GI Symposium on Parallel Computers and Parallel Mathematics, Munich, West Germany, March 1977.

Williams, A. C. and Bhandiwad, R. A. "A Generating Function Approach to Queueing Network Analysis of Multiprogrammed Computers." Networks (January 1976): pp 1-22.

Wood, David C. "A Survey of the Capabilities of 8 Packet Switching Networks." IFIP 1975 Computer Networks, pp. 1-7.

Wood, P. C. and Riviere, C. Programmable Front-End Processors Applied to Digital Data Communications Systems. Rome Air Development Center Technical Report TR-73-183, July 1973.

Wulf, W. et. al. "Overview of the Hydra Operating System Development." ACM Operating Systems Review, Vol. 9, No. 5, 19-21, November 1975, pp. 122-132.

Wulf, W. and Levin, R. "A Local Network." Datamation, February 1975, pp. 47-50.

BIOGRAPHY

Robert L. Leech was born in Chicago, Illinois on October 21, 1938. He graduated with a B.S. from the United States Military Academy at West Point, New York in 1960 and was commissioned a Second Lieutenant in the Regular Army as a member of the Signal Corps. He subsequently graduated from Purdue University in 1966 with the M.S.E. degree. He studied Computer Design at Columbia University in 1968-1969 while holding an appointment as Assistant Professor, U.S.M.A. from 1966-1969. He received an A.M. in Computer Science from Duke University in 1976. The author has been on continuous active duty in the U.S. Army since his commissioning. He currently holds the rank of Lieutenant Colonel and is a member of the Army Computer Specialty Field. He has graduated from the Basic, Advanced and ADP Career service schools and has held various Army Command and high level international staff assignments dealing with combat communications systems and the design, installation and operation of large scale computer systems as part of the Department of Defense Worldwide Military Command and Control System. His assignment locations have included the U.S., Vietnam and Europe. His current assignment is as Associate Professor, Department of Electrical Engineering, U.S.M.A. Military awards and decorations include the Bronze Star Medal with Oak Leaf Cluster, Meritorious Service Medal with Oak Leaf Cluster, Air Medal, Army Commendation Medal with Oak Leaf Cluster and Parachutist Badge. He is a registered Professional Engineer in Connecticut and is a member of ACM and the IEEE.

9 Aug 1977

Addendum
to
EQN Models for the Analysis & Design
of a
Computer Network of Functionalized Processors

Appendix A:

1. In procedure SOLVE the second EDIT statement after NOMORE must have a semicolon inserted (;) to read:
   F (15, 6));

2. In subroutine CHOOSE the following lines of code need to be inserted after "/* THIS IS AN ELIGIBLE ROW*/":

```
DO J=1 TO N-1;
IF TOP(J) > 'M' THEN
/* THIS IS THE BEST PIVOT SO FAR*/
```

p. 75: Eq. IV-43 should read Tr(1) = (|M|-1)m.